INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

LongLiveNoC: Wear Levelling, Write Reduction and Selective VC allocation for Long lasting Dark Silicon aware NoC Interconnects

by

Khushboo Rani

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Department of Computer Science and Engineering

Under the supervision of

Prof. Hemangee K. Kapoor

January 2021

Declaration of Authorship

I, Khushboo Rani, hereby confirm that:

- The work contained in this thesis is original and has been done by myself under the general supervision of my supervisor.
- This work has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to the authors/researchers by citing them in the text of the thesis and giving their details in the reference.
- Whenever I have quoted from the work of others, the source is always given.

Khushboo Rani
Research Scholar,
Department of CSE,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039,
khushboo@iitg.ac.in, kuchhu96@gmail.com

Date: Jan 29, 2021
Place: IIT Guwahati

Certificate

This is to certify that the thesis entitled **"LongLiveNoC: Wear Levelling, Write Reduction and Selective VC allocation for Long lasting Dark Silicon aware NoC Interconnects"** being submitted by **Ms. Khushboo Rani** to the department of *Computer science and Engineering, Indian Institute of Technology Guwahati*, is a record of bonafide research work under my supervision and is worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

Prof. Hemangee K. Kapoor
Department of CSE,
Indian Institute of Technology Guwahati,
Guwahati, Assam, INDIA 781039,
hemangee@iitg.ac.in

Date: Jan 29, 2021
Place: IIT Guwahati

Dedicated to

My loving parents

Your memories are eternal.

Acknowledgements

It is an immense pleasure to thank all the people who have supported me during my Ph.D. stay at IIT Guwahati. First and foremost, I would like to extend my sincere gratitude to my supervisor Prof. Hemangee K. Kapoor, for introducing me to this exciting field of Architecture and her guidance, encouragement, and extensive help over the last few years. Her enthusiasm and focus on research have always motivated me. She has given me the freedom to pursue research ideas and develop my research skills. I profusely thank her for correcting my silly mistakes again and again and keeping me engaged in my Ph.D.

I am thankful to my Doctoral Committee Members: Prof. Diganta Goswami, Dr. Aryabartta Sahu, Dr. Santosh Biswas, Dr. Arnab Sarkar, and Dr. Chandan Karfa for their productive and constructive suggestions for my thesis work. I firmly believe that their opinions and comments help me to shape up my final thesis. Additionally, my sincere thanks to Prof. S. V. Rao and Prof. Jatin K. Deka, the Head of the Department of Computer Science and Engineering, and other department faculty members for their constant support and helps. Furthermore, my sincere thanks to Prof. Kalpesh Kapoor for various academic/non-academic suggestions. I would also like to thank Dr. V. Vijaya Saradhi for giving all the supports during TA duty. I must thank Prof. Prabhat Mishra (Director, Embedded Systems Lab, UFL) for encouraging me towards quality work and publication.

During my Ph.D., I got the opportunity to work with Dr. Shounak Chakraborty, Dr. Sukarn Agarwal, Palash Das, Sheel Sindhu Manohar, and Arijit Nath. For countless times, sharing knowledge and fruitful technical discussions with them helps me carry out my research work. Various academic and non-academic discussions were always fun with them. I especially thank Dr. Sukarn Agarwal for his initial guidance in the computer architecture simulators and the research.

My friends and well-wishers have greatly helped me, and without their help, my work would not have been possible. I feel lucky to have a company of Dr. Shounak, Dr. Jitendra, Dr. Sukarn, Durgesh, Dr. Satish, Dr. Satisha, Pallabi, Palash, Arijit, Sheel, and Shreshtha (all PhDs). An uncountable number of random discussions with them makes my life joyful. My thanks are also due for Kiran, Venkatesh, Shreya, Vijay, and many more for their tremendous support, which few people will get.

I sincerely thank Mr. Raktajit Pathak, Mr. Nanu Alan Kachari, Mr. Bhriguraj Borah, Mr. Monojit Bhattacharjee, Mrs. Gauri Deori and all other department staff members for helping me different ways and at different times during my stay at IIT Guwahati. I would also like to thanks the student affairs section for providing an on-campus hostel facility. Last but not least, I am conveying my appreciation to security guards, janitors, hostel mess, and stationary and canteen staff for making my life smooth on the IITG campus.

Above all, I am incredibly fortunate to have moral support and encouragement from my parents, who believed in me and guided me towards the right path for their whole life. I can not thank enough for the incredible life they have given to me. Their infallible love and support have always been my strength. Their patience and sacrifice will remain my inspiration throughout my life. I am extremely thankful to my brother,Anjani Anjan, and his wife, Reeshma Ramesan, to encourage me during all ups and downs of my personal and Ph.D. life. I am also very much grateful to all my family members for their constant inspiration and encouragement.

Last but not least, I would like to thank those who discourage, demoralize, and menace me at different life stages. All these motivated me to take my failure as a challenge and work hard to succeed.

Abstract

With the continuing advancement in CMOS technology, we have more and more transistors packed on the same die leading to having multiple processing cores on the chip known as Chip Multi Processors (CMPs). Communication across multiple cores happens with the help of the switch-based network-on-chip (NoC). The design space of NoC is tightly constrained as the chip real estate needs to be distributed across the multiple cores as well as multiple levels of caches. The same constraint applies to power consumption. It has been noticed that communication consumes almost 36% of the total chip power. With tighter power budgets and to meet the thermal design power (TDP) for the system, components like the cores/-caches undergo voltage and frequency scaling and, at times, power off. Powering off several components to stay within the TDP leads to the concept of dark silicon. In dark silicon, although the cores/caches are off, the communication network is expected to be available. In order to reduce the standby power of the network in such events, one looks for avenues in non-volatile memory (NVM) technologies. Attempts have been made using a combination of NVM and SRAM buffers in the routers for energy efficiency. Despite many advantages, the NVM buffers suffer from costly write operation, and weak write endurance. The challenge in this type of work is the effective utilization of the NVM buffers.

According to the available literature, due to low write endurance, the lifetime of NVM buffers is limited. Additionally, the application behaviour at the run-time and virtual channel (VC) allocation policy lead to write-variations across VCs. In that, some VCs and virtual networks (VNets) get written heavily compared to others, which are termed as Intra-VNet, and Inter-VNet write-variation, respectively. To mitigate these unwanted write-variation and achieve better lifetime of NVM based router buffers, wear-levelling techniques have been proposed in this thesis. Towards this, we propose wear-levelling with static and dynamic buffer allocation to VCs. In the static buffer allocation technique, the NVM buffers are statically mapped to VCs, and the writes are distributed evenly to reduce write variation in a VNet. We have considered two scenarios: iso-capacity and iso-area, and proposed VC allocation policies for both since the number of buffers differs in both the scenarios. From this contribution, we get around 99% reduction of write-variation over basic VC allocation policy, which results in significant improvement in the lifetime of buffers. The dynamic buffer allocation policies further enhance

the lifetime of NVM buffers by reducing write variation across all VNets. By experimental analysis, we achieved around 20 times lifetime improvement over basic VC allocation policy.

In the dark silicon scenario, the routers connected to dark processing elements get very less traffic passing through it. Hence, the power consumption in such routers can be optimized by using frequency scaling technique or by keeping only NVM buffers active. From this, we get a 46% reduction in total energy and ten times lifetime improvement over baseline design.

Another challenge for employing NVM as a router buffer is the costly write operation. Towards minimization of the effect of the costly write operations of NVM buffers, with the consideration of performance and energy as a major system-wide constraint, we proposed write-reduction techniques. The first approach compresses an incoming packet to the network by removing zero-words. To maintain the location of such words, a bit-vector is used. At the destination, the packet gets reconstructed before it gets forwarded to the protocol buffer. This approach reduces flits in the network by 48% and enhances lifetime by seven times. The second approach reduces packet size by removing clean flits during the cache-write back. This reduces network flit by 27% and no packet latency overhead is observed over baseline design.

The thesis has thus demonstrated effective management techniques for writes in NVM buffers for an optimal lifetime and controlling the effect of costly write operations.

Contents

List of Figures

List of Tables

Abbreviations

CMP	Chip Multi-Processor
TCMP	Tile Chip Multi-Processor
NoC	Network on Chip
IPC	Instruction Per Cycles
CPI	Cycles Per Instruction
LLC	Last Level Cache
SRAM	Static Random Access Memory
DRAM	Dynamic Random Access Memory
NVM	Non-Volatile Memory
MRAM	Magnetic Random Access Memory
STT-RAM (or STT)	Spin Transfer Torque Random Access Memory
PCRAM (or PRAM)	Phase Change Random Access Memory
ReRAM (or RRAM)	Resistive Random Access Memory
MPKI	Miss Per Kilo (Thousand) Instructions
NUCA	Non Uniform Cache Access
UCA	Uniform Cache Access
SNUCA	Static NUCA
DAM	Dynamic Associativity Management
EDP	Energy Delay Product (Energy $\times$ Delay)
VC	Virtual Channel
VNet	Virtual Network
IC-Base	Iso-Capacity Baseline
IA-Base	Iso-Area Baseline
Intra-VNet	Write variation across VCs in a Virtual Network
Inter-VNet	Write variation across Virtual Networks
WVAR	Write VARiation aware VC allocation
Hy-WVAR	Hybrid Write VARiation aware VC allocation
SET-RR	SET Round Robin
SET-VNET-WC	SET Virtual NEtwork Write Count

SET-VNET-WC-WVAR	SET Virtual NEtwork Write Count with WVAR
SET-VNET-WC-Hy-WVAR	SET Virtual NEtwork Write Count with Hy-WVAR
Hy-Drowsy	Hybrid Drowsy policy [1]
Dy-WVAR	Dynamic Write VARiation aware VC allocation
Hy-Dy-WVAR	Hybrid Dynamic Write VARiation aware VC allocation
bg	Buffer group
gBUF-DN	Gated Buffers of Dark Nodes
SRAM-FRQSCL	Frequency scaling in SRAM buffer router
Hy-SEL-ON	Hybrid Selective ON
Hy-NEIG-SEL-ON	Hybrid Neighbour Selective ON
Hy-CRIT-WRD	Hybrid Critical Word
NIC	Network Interface Controller
ZENCO	Zero byte based ENCOding technique
ZENCO_RR	ZENCO Round Robin
DidaSel	Dirty Data based VC Selection

Chapter 1

Introduction

Moore's Law [13, 14, 15] suggests that the number of transistors in an integrated circuit doubles approximately every two years. This drives architects to increase the core counts in order to exploit Moore's Law scaling, rather than focusing on improving the single-core performance. As shown in Figure 1.1, the transistor count on an integrated circuit (IC) doubles every two years. Accordingly, by the year of 2012, the state-of-the-art processors already contain billions of transistors (such as the Intel's 7.08 billion transistors GPU [13]). The technology scaling leads to more processing core units, and large caches on the same die called the Chip Multi Processors (CMPs). The setup of multiple cores and caches on a chip can function efficiently only in the presence of an equally efficient on-chip interconnect since the traditional bus-based designs are no longer scalable. The packet-based network-on-chip (NoC) provides a scalable and efficient design choice alternative for on-chip communication for cores and caches.

1.1 Modern Chip Multi-Processors (CMPs)

In CMPs, the large number of cores are integrated on the same die, with identical (homogeneous) or different (heterogeneous) power budgets. The cores employed in the CMP are either complex or simple superscalar processors that can perform

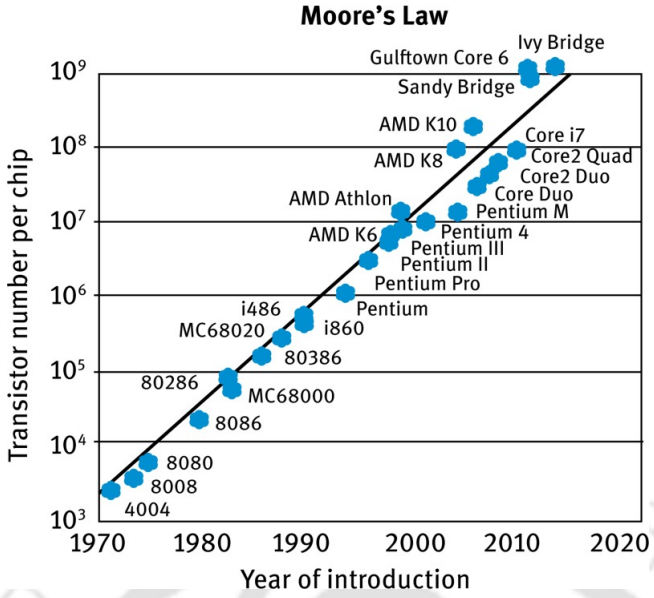
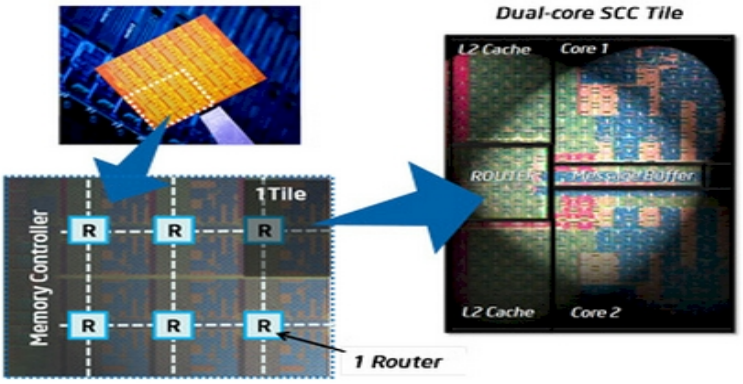


FIGURE 1.1: The plots of transistor counts against dates and Moore's Law (courtesy of [4])

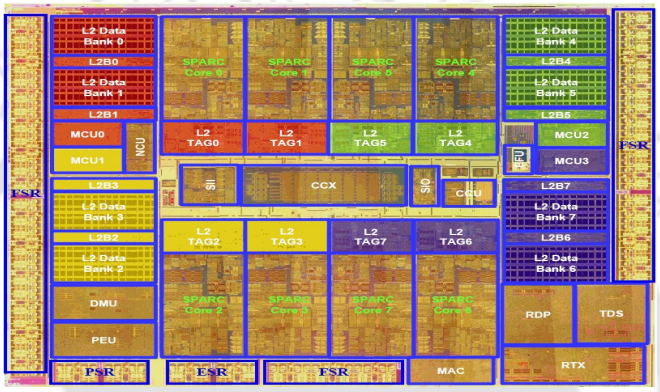
a variety of tasks. The main goal to design CMPs were to improve the performance and to add the parallelism. Additionally, the performance of a CMP can be enhanced further by integrating more number of identical cores in each new generation. With time, upcoming generation workloads and applications need higher throughput and parallelism with the larger data demands. The multi-level on-chip caches are attached to fulfil the data demands. Whereas to achieve parallelism, the CMPs becomes larger, and it needs a strong communication infrastructure which can only be provided by the modern communication system, called Network-on-Chip (NoC). Figure 1.2 presents the design and the floor-plan outlines of two modern CMPs architectures: Intel Single-Chip Cloud Computer [16] and UltraSparc T2 [18].

The modern CMPs are built by using the following components:

1. The processing cores as the computation unit, CPU cores.
2. An on-chip storage to retain most of the required data needed by the running application, On-chip Caches.
3. The communication medium through which on-chip communication between different component happens, Network-on-Chip.



(a) Intel Single-Chip Cloud Computer - A 48-core processor [16]



(b) UltraSPARC T2 [17]

FIGURE 1.2: Modern CMPs: Design and Floor-plan outlines

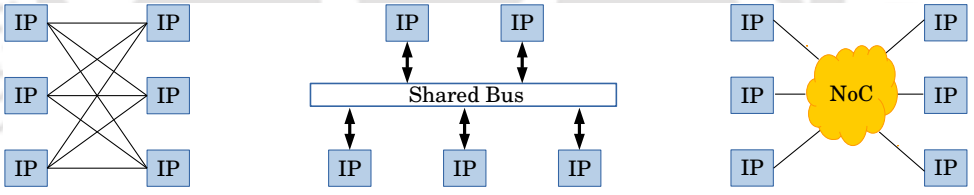


FIGURE 1.3: The trend of on-chip interconnections. (a) Point-to-Point interconnect (b) Shared Bus based system (c) Network on Chip (NoC) based system

An efficient way to manage communication among the on-chip resources has become critically important in modern CMPs. Figure 1.3 shows the trend of on-chip interconnection. The peer-to-peer interconnection consumes large wire area which results in a large area of the chip [19]. The bus-based architecture suffers from limited bandwidth [19]. Both the approaches suffer from scalability issues which brings the power and transmission delay overhead when chip size reduces beyond 45nm. For the efficient use of hundreds or thousands processor elements (PEs)

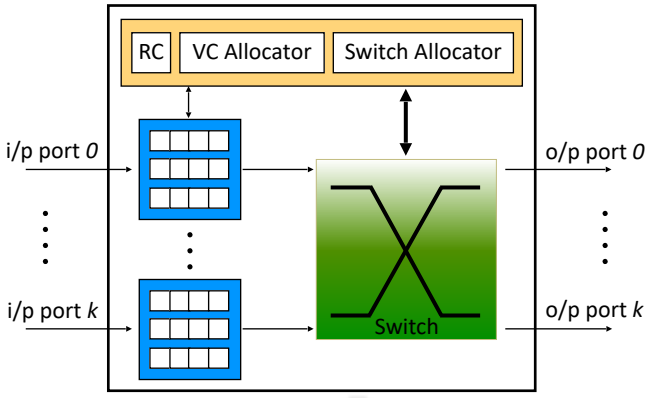


FIGURE 1.4: Basic Router Architecture

in CMPs, Network-on-Chip has been proposed as an efficient and scalable solution. Similar to the concepts in internet and wireless network, NoC uses routers to route packets instead of wires [20]. The latency and throughput performance is improved due to the higher network bandwidth. Also, the power consumption can be significantly reduced by breaking long links between the processors and avoiding high fan-outs in the outputs. The NoC provides more scalable communication than the other two schemes. Hence, for the technology nodes beyond 50nm, NoCs are more preferred over the other two paradigms.

1.2 NoC Architecture in CMPs

Figure 1.4 depicts the micro-architecture of a general-purpose NoC router. The NoC router mainly has five stages: Buffer Write (BW), Router Computer (RC), Virtual Channel Allocation (VA), Switch Allocation (SA) and Switch Traversal (ST). The flits go through all these stages inside a router, and after switch traversal, it traverses the link between the current and downstream router.

Each port uses wormhole routing and has multiple VCs per input port. Credit-based flow control is used to avoid buffer overflow and also guarantees free buffer space at the downstream input port. This also controls flit transmission and buffer overflow as well as packet-loss. The buffers in NoC are very simple and mostly FIFO based, having to say k number of entries. These buffers are managed using head and tail pointers so that the flits do not need to traverse all the buffers

before exiting. VCs are logically grouped under a VNet. Packet/flit travels from one router to another over one virtual network to avoid protocol-level deadlock. VC allocator assigns the packet to a VC belonging to the appropriate class (i.e. VNet), similar to the one of the downstream router. Within the VNet the VC allocator looks for a free/available VC to be assigned to the packet. As the flits are output, the VCs become available to be used by the subsequent packets.

Flits coming to the router get written into the input buffer in a respective virtual network (VNet) and VC. The Header flit contains the information about the VC and VNet which was decided in the upstream router. Next, in the route computation, the output port is calculated based on the destination router address and routing protocol. Once the output port is decided, the packets go through the VC allocation stage. The Switch allocator finds winner among all the requests from the input ports for the output ports. There can be a maximum of one winner per output port. The winning input port and the VC gets full control of switch traversal in the next cycle, and the packet/flit gets written to the output buffer. The RC and VA stages perform the computation for the head flit only (once per packet). Whereas, the body flits pass through these control stages with no computation. The SA and ST stages operate on every flit of the packet.

1.3 Power Consumption in CMPs

For each of the on-chip components, the total power consumption can be divided into three major parts [21, 22, 23]: (1) Dynamic Power, (2) Static Power and (3) Short Circuit Power.

1.3.1 Dynamic Power

Dynamic Power is defined as the power consumed due to the On-Chip switching activity of the transistors and circuitry due to the charging and discharging of capacitance. In general the dynamic power is the processing power which is

represented by the help of following equations [24, 23]:

$$P_{Dyn} = \alpha.C.V^2.f \tag{1.1}$$

In the eq. 1.1, P_{Dyn} represents the dynamic power and α, C, V and f denote activity factor, capacitance, supply voltage and running frequency of the component, respectively. As can be seen, the power consumption is directly proportional to its operating frequency.

In NoC, the major portion of dynamic power is consumed by two basic units: (1) Routers and (2) Links. The routers consume most of the dynamic power in NoC. The router's components responsible for dynamic power consumption are [25]:

1. **Router Clock:** An essential part of the router which is responsible for maintaining synchronisation.
2. **FIFO Buffers:** The buffers store the incoming flits (part of a data block) coming at the input port of a router. The flit stays in the input buffer until it gets written into the output buffer after passing through all the control stages.
3. **Arbiters and Allocators:** These make sure that the data block/packet reach the proposed destination.

The detailed modelling of the power consumption for different types of memory technologies used for the fabrication is discussed in Chapter 2.

1.3.2 Static Power

The power drawn by the on-chip circuitry even when the circuitry is not performing any task is defined as the static power. The static power mainly indicates the circuit's leakage power which depends on two important components: (1) Subthreshold Leakage and (2) Gate Leakage [24, 23, 26, 27]. Between these two units,

the sub-threshold leakage has a direct dependency with the chip temperature and supply voltage. As chip temperature increases, the covalent bonds in the atoms of semiconductor materials break. This results in the release of electrons which starts flowing through the reverse bias and generates current, called sub-threshold leakage current. The power consumed due to the Sub-threshold Leakage current is known as Sub-threshold Leakage power. The Gate Leakage power consumption happens due to the down-scaling device size and the reduced thickness of gate-oxide materials.

The relation of sub-threshold leakage with running temperature and supply voltage is given in following equation [28, 24]:

$$P_{Stat} = K_1 V_{DD} T^2 e^{((\alpha V_{DD}+\beta)/T)} + K_2 e^{(\gamma V_{DD}+\delta)} \tag{1.2}$$

In the eq. 1.2, P_{Stat} represents the static power consumption due to sub-threshold leakage for a CMOS circuit. V_{DD} is the supply voltage and T denotes the current temperature. K_1, K_2, α, β, γ and δ are empirical constants that depend on the circuit size, topology, technology and design.

Total static power of a CMP comes from the static power consumed by all of its components, i.e. cores, caches and NoC.

1.3.3 Short-Circuit Power

The short circuit power is consumed due to non-zero rise/fall time of the CMOS circuitry. The power is consumed during a short time-span when both NMOS and PMOS are active simultaneously, which makes a direct link between supply voltage and ground. In general, this power consumption is negligible and often ignored while analysing power consumption of modern CMPs [24, 23, 26].

This dissertation considers only the static and dynamic power/energy for the calculation of power/energy consumption.

In the case of modern CMPs, among all the on-chip components, cores are usually accounted for their high dynamic power consumption whereas caches and on-chip interconnects are considered for their high leakage energy. Also, in modern CMPs, the number of transistors increases and occupies large wafer real estate. This leads to the large leakage power consumption, which is a major contributor to the total power consumption of the chip. This motivates computer architects to reduce the power consumption due to on-chip interconnects. The thesis focuses on reducing the leakage power consumption of the NoC routers by employing the emerging near-zero leakage power NVM buffers.

1.4 Motivation

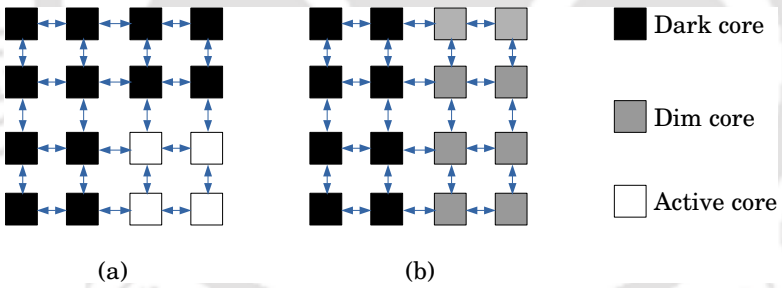


FIGURE 1.5: Different application mappings in Dark Silicon scenario.

With technology scaling, we have more and more transistors available on the chip. Dennard scaling [29] claimed that the performance per watt of computing grows exponentially at roughly the same rate. It suggested that even if the technology scaled, the power density remained constant, and hence, the power dissipation stayed in proportion with the area. However, with recent sub-micron technology scaling, Dennard Scaling no longer holds, and the power dissipation increases, leading to the rise in chip temperatures. Scaling of technology leads to challenges

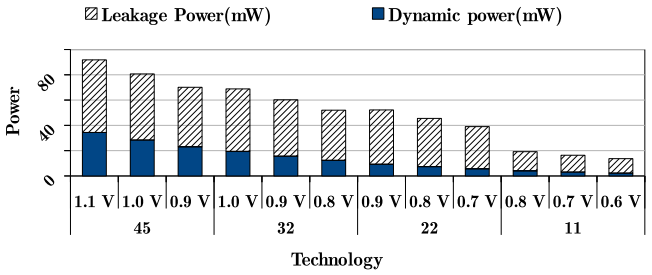


FIGURE 1.6: Static and dynamic power breakup of a router at different technology nodes.

related to increased power dissipation, mainly due to the leakage currents. To mitigate the leakage power dissipation and to meet the thermal design power (TDP) limit for the system, few components, like the cores and caches, undergo voltage and frequency scaling and, at times, power off. Powering off several components to stay within the TDP leads to the concept of dark silicon [30, 31]. With further scaling in technology and more integration of cores on the chip, it is predicted that the dark area will range from 20 to 80% of the chip [32, 33]. ITRS suggests that by 2020 only 10% of the chip's hardware resources are useful at any given time due to the on-chip power density [34]. A comprehensive discussion of the state-of-the-art approaches for dark silicon can be found in [35, 36]. Figure 1.5 shows two dark silicon patterns in 4×4 TCMP, where fig. (a) has 75% of dark cores, and the remaining four cores are running on full frequency. Whereas in fig. (b) 50% of cores are kept dark and remaining 50% cores are running on low power mode, making it dim. Although the cores and or caches are dark/power-gated or grey/dim (operating at low power mode), the communication network is expected to be available in dark-silicon [37].

1.4.1 NoC Power Problem and Connectivity

The 16-tile MIT RAW On-Chip network consumes 36% of total chip power, with each router dissipating 40% of individual tile power [38]. Whereas, in Alpha 21364 microprocessor, router, and the links consume approximately 20% of the total chip power [39]. The router presented in [40], consumes 28% of tile power in comparison to tiny cores. In general, the NoC occupies a significant area on

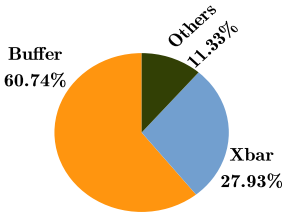


FIGURE 1.7: Dynamic Power distribution across various router components.

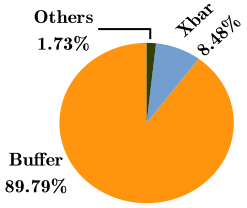


FIGURE 1.8: Static Power distribution across various router components.

the wafer real-estate and consumes a significant amount of total chip power over which leakage power becomes the major contributor [41] [42]. As can be seen from the figure 1.6, the leakage component of the power budget is significant compared to the dynamic power, and the share of leakage increases as technology scales. The figure 1.7 and 1.8 depict dynamic and leakage power breakdown for router components. The network buffer is responsible for the majority of leakage power dissipation in a router. It shows that in dark silicon setup, when a majority of cores/caches are power-gated, NoC (and mainly its buffers) may become a major power contributor due to leakage power consumption.

In the interest of reducing the standby power of the NoC in dark silicon, Non-Volatile Memory (NVM) technologies can be used as an alternative for the router buffers.

The chip designers are considering recent emerging Non-Volatile Memory (NVM) technologies as a design alternative of conventional charge based volatile memory technologies. There are several types of NVMs that are under research, out of which some have reached advanced development stages and are commercially manufactured, and some are not yet mature. One of the states of art [6] lists 12 such NVM technologies: Spin Transfer Torque Random Access Memory (STT-RAM), Phase Change Random Access Memory (PCRAM), Resistive Random Access Memory (ReRAM), etc. We limit our ideas to the STT-RAM [43] as it is extensively studied and considered as a viable replacement to conventional charge based volatile memory technologies like SRAM. Compared to the SRAM, the NVM

technologies use resistance as an information carrier. This makes them prone to soft reliability errors.

NVM technologies offer many advantages over conventional SRAM technology. These advantages include large retention time or non-volatility, high density, low standby leakage power, multi-bit storage capability, and read latency and energy compared to conventional counterparts. Despite many advantages, there are some drawbacks that reduce the chances of NVMs to be used in the actual hardware. The disadvantages include large write latency and dynamic energy due to the long time and large energy consumption for bit flipping. To mitigate the expensive write operations, researchers have used hybrid memory technology to make use of the best characteristics of SRAM and NVM technologies [1] [44]. Another leading set of a problem with the employment of NVM buffers arises due to the weak write endurance [8] (the ability of NVM cell to withstand a certain number of writes before it breaks out). The write endurance value of these NVMs are 10^8 writes for PCRAM, 10^{11} writes for ReRAM and, for STT-RAM the predicted value is 10^{15} writes but the write endurance value tested so far is 4×10^{12} writes [8, 45, 46, 47]. Furthermore, the weak write endurance of the NVM buffers can also get affected by the unwanted write variations generated by the system. In other words, the writes to the router buffers are non-uniformly distributed among virtual networks, i.e. Inter-VNet and among virtual channels inside the virtual network, i.e. Intra-VNet. Due to the non-uniform write distribution across buffers, there are certain VCs with more number of writes than others. If we design NoC buffers using STT-RAM, then the number of writes to them will be governed by the NoC policies, and this variation will affect the lifetime of the individual buffers: as some buffers may get written several times more compared to some other buffers. This will indirectly affect the lifetime of the router as a whole. These above mentioned circumstances encourage us to develop techniques to reduces costly write operations and maintain uniform writes to improve NVM lifetime.

1.5 Thesis objectives

The main aim of my research work is to reduce the leakage power consumption of NoC router buffers. Towards achieving this, we propose to use NVM based buffers. However, NVM buffers suffer from weaker write endurance and costly write operation. To overcome the shortcomings of NVM buffers, we list four objectives of this thesis which are as follows:

1. Minimize write variation across the virtual channels in each VNets
2. Minimize write variation across the VNets in the input port buffers
3. In Dark Silicon, when most of the nodes are power-off, identify methods to save energy and improve performance while maintaining connectivity
4. Reduce the number of writes in NVM buffers to reduce costly write operation and improve the lifetime

1.6 Thesis Contributions

The major contributions of this thesis are be summarised below:

1.6.1 Wear Levelling by Static Buffer Assignment to VCs

The objective of the thesis is to employ NVM technology-based buffers in the router of the NoC for their energy efficiency and use them effectively by taking care of wear levelling and write variations. In the first contribution, we assign a set of buffers to each virtual channel, and these buffers remain permanently assigned to the particular VC throughout the lifetime of the design. The contributions will then involve the allocation of incoming packets to particular VCs while taking care of wear levelling. As STT-RAM is 3-4 times denser than SRAM, we have conservatively used three times the number of SRAM buffers for our STT-RAM

counterparts. Write variation aware VC allocation to packets can be done in two scenarios: Iso-Capacity and Iso-Area.

1.6.1.1 Iso-Capacity based VC allocation: WVAR, Hy-WVAR

Here we assume the same number of VCs as in the baseline SRAM based NoC router. These VCs can be made using buffers made of SRAM or STT-RAM. In WVAR all buffers/VCs are STT-RAM based while in Hy-WVAR there is a combination of STT-RAM buffer based VCs and SRAM buffer based VCs. In order to uniformly distribute the writes, we maintain counters with each STT-RAM based VC. When a write happens to a VC buffer, the counter associated gets incremented. These write counts for all VCs within a VNet, are used to uniformly distribute the writes. In particular, during the VC allocation stage, among the free STT based VCs, the VC with minimum write counts is allocated to the packet. The usage of STT-RAM buffers increases the packet latency. To overcome this limitation, we propose the Hy-WVAR policy which uses an alternative SRAM VC, as SRAM has lesser write latency. When the network traffic increases beyond a predefined threshold, VC allocator activates SRAM VC in place of one heavily written free STT-RAM VC.

1.6.1.2 Iso-Area based VC allocation: SET-RR, SET-VNet-WC

In this work, we utilise the fact that STT-RAM has more density than SRAM. On the same chip area, the capacity of STT-RAM can be around 3-4 times the capacity of SRAM. We use this fact and (conservatively) take STT-RAM buffers 3-times the number of SRAM buffers. Having additional buffers can help in enhancing the lifetime by utilising them judiciously. The extra buffers are used in forming additional virtual channels and then virtual networks. We then select a subset of these VNets while running the application. In case we allow all the VNets to be available for use, then the write variation increases. Therefore, the available VNets are divided into (three) sets, and we use VNets in one set over an interval before

switching to another set during the next interval. The policy is called SET-RR, and it distributes the writes evenly in the VCs in the sets.

The above policy reduced the write variation but not completely. This is because, across the sets, there is still some variation. SET-RR selected a VNet from a particular set and used it throughout the interval. However, it may happen that certain VNets within one set have incurred more writes compared to VNets in another set. Therefore, at the start of each interval, we propose to select VNets from different sets depending on their write counts.

In both SET-RR and SET-VNet-WC, once a VNet is selected, WVAR allocation policy is applied. Also, to improve the write latency, Hy-WVAR variant is also implemented and evaluated.

This work is fully discussed in Chapter 3.

1.6.2 Wear Levelling by Dynamic Buffer Assignment to VCs: Dy-WVAR, Hy-Dy-WVAR

In the second contribution, we assign a group of buffers to each virtual channel over an interval. However, at the end of interval depending on the write counts, a buffer group may get assigned to another VC. In particular, a VC is logically constructed by buffer-groups, and after each interval, these buffer groups are reassigned guaranteeing that the writes in the buffer-groups are evenly distributed. In particular, buffer groups having more write counts are allocated to VNets incurring less write counts. This helps in balancing the number of writes and reducing write variation. Note that inside a buffer group, static allocation policy is applied.

This detailed description of this work is given in Chapter 5.

1.6.3 Power saving using Frequency Scaling and Power Gating

Our next contribution attempts to improve the lifetime of NVM buffers by reducing the number of writes in them and save energy by powering off some components. In this work, we exploit the fact that buffers consume maximum power and attempt to save power at the level of buffers. We keep the routers always powered-ON to keep constant connectivity in the dark silicon. To save leakage power, in this work, we propose five policies and investigate their impact on the performance. The various proposals include power gating, frequency scaling, using hybrid buffers and powering off some of them.

1.6.3.1 Gated Buffers of Dark Nodes: gBUF-DN

In the context of dark-silicon, most of the processing elements (PE) will be powered off. In such a scenario, one needs to decide the status of the routers associated with such powered off tiles from the energy consumption and connectivity aspect. In order to save energy, most of the router buffers can be powered off and to maintain connectivity, only a minimum number of buffers are kept powered on in each VNet. Once the PE is turned on, all the router buffers are powered on.

1.6.3.2 Frequency scaling of SRAM based buffers: SRAM-FRQSCL

Frequency scaling is done for routers connected to powered-off PEs to save leakage energy. Initially, the application runs with all routers at maximum frequency. Subsequently, at regular intervals, the routers undergo frequency scaling if the PE is off. Once the frequency is scaled, it remains so throughout the interval and can be changed only at the end of the interval (if the PE is turned-on in the meantime). For our experimental evaluation, we propose to reduce the frequency to half of the maximum frequency.

1.6.3.3 Selective power gating of Hybrid buffers: Hy-SEL-ON

The above two proposals use all SRAM based buffers (or VCs) and either power gate buffers or reduce the frequency of the router to save power. In this policy, a few SRAM based VCs are replaced with STT-RAM based VCs, as STT-RAM has very little leakage. In this case, if the attached PE is powered-off, then all the SRAM based VCs will be powered-off, and the traffic (which is expected to be less) will be handled by the STT-RAM VCs. The procedure is invoked at regular intervals, and the decision taken at the beginning of the interval is maintained throughout the interval. A write variation aware VC allocation is done while using STT-RAM VCs to improve the lifetime.

1.6.3.4 Selective power gating of Neighbouring Hybrid buffers: Hy-NEIG-SEL-ON

For the routers that have PE powered-off, there is expected to be very less traffic passing through them. This is also true for the connected ports of the immediate neighbouring nodes. For instance, if the central router has powered-off PE, then the VCs at input ports have STT-RAM VCs being used. As traffic coming to these input ports from neighbours may also be less, we propose to save the leakage of the neighbouring routers by making their output port VCs behave similar to the input port VCs of the powered OFF node.

1.6.3.5 Prioritizing Critical Words in Hybrid buffers: Hy-CRIT-WRD

The idea of using STT-RAM buffers for routers associated with active nodes increases network latency. This causes an increase in network latency over SRAM based buffer designs. The delay, caused due to the usage of STT-RAM buffers, in critical packets may impact the overall performance of the system. To send critical words faster, the policy Hy-CRIT-WRD uses SRAM based VCs for critical words (or packets) even when there are only STT-RAM VCs active. At the time of VC allocation, critical word (or packets) gets assigned to SRAM based VC. For

the active routers, the SRAM VCs are already active; hence the critical packet is allocated to it. Whereas for the routers with only STT-RAM VCs active, an SRAM VC is woken up and kept active for a predefined number of cycles. This reduces the network latency for the critical words (or packets), which improves system performance for the critical word optimisation.

A more detailed description of this work is given in Chapter 5.

1.6.4 Write Reduction Techniques

The earlier contribution proposed wear levelling approaches to improve the lifetime of NVM buffers. Lifetime depends on the number of writes incurred by the buffers. If we reduce these writes, we can also improve lifetime. The first proposal is based on compression, where the data is wrapped in a more compact way before it is transmitted or stored. The second proposal is based on dirty data sent by the cache during write-backs. The identification and transmission of only the modified content within a cache block during write-backs can limit the writes in the network.

1.6.4.1 Zero-bytes based ENCOding: ZENCO

There is a considerable percentage of cache blocks that either has only zero bytes or more than 50% of the bytes as zero. Removing these zero bytes from the cache block (data packet) coming to NI will reduce packet size significantly, which will result in lesser flits in a network. In this technique, zero bytes are taken into consideration when performing the compression. If the zero words are identified in the data packet, then it is not transmitted in the network. A binary array (say isZeroVector) is maintained to keep the entry of each word of the uncompressed data packet. The isZeroVector array is used at the decompressor to re-generate the original compressed data packet.

1.6.4.2 Dirty data based Selection of VC: DidaSel

In this work, a flit based write reduction technique is proposed to save the writes in STT-RAM buffers, which is based on dirty flits present in data packets. Based on our experiments, 70% of cache blocks that travel through the network are not dirty. Limiting data-write packets to only dirty flits/words leads to lesser traffic in the network and lesser writes in NVM buffers. This saves dynamic energy and reduces the overall network latency of the packets.

The proposal uses hybrid buffers. When complete packets need to be sent over the network, then SRMA buffers/VCs are used. When only dirty-flits of a packet are to be sent, then these are candidates to be sent over STT-RAM VCs. The decision is based on a threshold value. If the number of dirty flits is below the threshold, then send using STT-RAM VC otherwise using SRAM VC. This reduces the number of writes taking place in STT-RAM buffers, thus increasing their lifetime. In case the chosen type of VC is not free, then the flits are sent via the first available VC irrespective of its type.

The full description of this work is given in Chapter 6.

1.7 Summary

In the context of dark silicon, several components (cores/memory) are kept powered-off to meet the thermal design constraints of the chip. NoC is the backbone of communication for today's chip multi-processors and also a significant contributor to the power budget. The buffers made up of SRAM suffer from high leakage power and occupy a large area on the wafer real-estate. Emerging Non-Volatile Memory (NVM) technologies are an attractive alternative for NoC buffers due to their low leakage power and high density. However, the major challenges with the employment of the NVMs are costly write energy and latency and, the lower lifetime because of the write variation and weak write endurance.

In our first work, we deal with the low write endurance problem of NVM buffers. Towards this, we proposed iso-capacity and iso-area-based VC allocation policies to reduce write variation across buffers. Both these proposals perform VC allocation to new packets based on their write counts, so as to evenly distribute the writes across all VCs. All proposals reduce write variation to almost 0% resulting in lifetime improvement in the range of 4-12 times for pure STT-RAM based VCs.

In the first contribution, the buffers are permanently assigned to form VCs. In the next proposal, we go at a finer granularity to control write variation. This is done by creating small groups of buffers and then assigning them to create VCs and VNets depending on the write counts across the buffer groups. With reducing write variation to almost 0%, we obtain a lifetime improvement in the range of 29 - 55 times over baseline STT-RAM design.

Our third contribution investigates avenues to reduce power consumption by selective power gating and frequency scaling. In the context of dark silicon, when most of the processing elements (PE) are power gated, the NoC is expected to maintain connectivity while saving energy. Our proposal of scaling frequency of buffers attached to power gated PEs gives energy savings of around 4% with degradation in packet latency by 8%. Using hybrid buffers at routers and using only STT-RAM buffers when the PE is power gated constitutes our other proposals. These lead to energy savings in the range of 43-45% with latency overheads in the range of 7-10 %.

The fourth contribution aims at reducing the number of writes happening in the STT-RAM based buffers so that the lifetime can improve. This is achieved by compressing packets with zero value content or by sending only dirty data flits instead of complete data packets during write-back. Both these proposals get compression ratios in the range of 0.37 to 0.52. The reduced network traffic achieves 19% dynamic energy savings with no impact of latency. The wear levelling variant of these achieves a lifetime enhancement of 7-14 times.

The thesis proposes ideas for better write management and proper VC allocation to packets in the NVM based/hybrid buffers. These ideas can make NVM as a better and reliable choice over SRAM to design the router buffers in dark silicon setup.

1.8 Thesis Organization

The thesis is organised as follows.

- Chapter 2 discusses the background details of different NVM technologies and motivations of our works. The prior literature works are also discussed in this chapter.
- Write Variation aware wear levelling approach using Static buffer Assignment to Virtual Channels is discussed in Chapter 3.
- Chapters 4 discusses the policies related to Dynamic Buffer Assignment to Virtual Channels to improve the lifetime of Non-Volatile buffers.
- Investigation of Frequency Scaling, Non-Volatile, and Hybrid Memory Technologies for On-Chip Routers in the context of dark-silicon is presented in Chapter 5.
- Write reduction techniques using zero bytes based compression and dirty data-based compression are given in Chapter 6.
- Finally, conclusion and future directions are given in Chapter 7.
- Appendix reports the experimental setup of all the works proposed in this thesis. This chapter discusses how the proposed methods are modelled (simulated) and compared with the other existing architectures.

Chapter 2

Background

As mentioned in Chapter 1, NoC in modern CMPs consumes significant On-Chip power and the router buffers contribute to a major portion of this power. The leakage component of the power budget is significant in the router power compared to the dynamic power, and the share of leakage increases with the scaling of technology. Power dissipation is linked to an increase in chip temperature. To meet the Thermal Design Power (TDP) budget for the CMP, dark silicon has emerged as a promising paradigm. Here several of the components are powered down to save leakage. In the presence of dark silicon, the processing cores are turned off when not in use; however, the on-chip interconnect is expected to be available to maintain connectivity. In other words, the routers connected to powered down processing elements are kept on.

The main aim of this thesis is to design an energy and packet latency efficient NoC router architecture by considering performance as a system-wide constraint. Hence, before discussing state-of-the-art router designs developed over the decade, we initially discuss some preliminary concepts/results regarding router micro-architecture, relevant to our work.

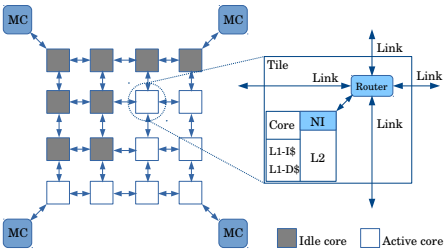


FIGURE 2.1: Tiled CMP architecture

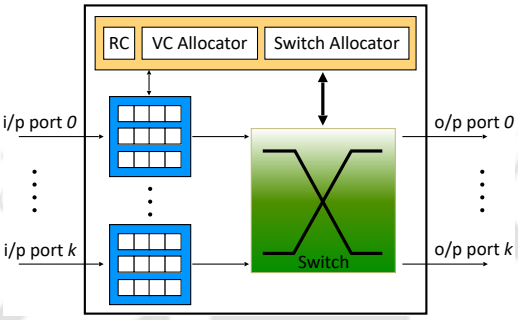


FIGURE 2.2: Basic Router Architecture

2.1 TCMP and Basic Router Architecture

The chip multiple processor setup used in this thesis is a Tiled chip-multiprocessor (TCMPs). This has evolved as a scalable design for small-scale future CMPs. A TCMP, shown in figure 2.1, consists of processing cores arranged in a mesh network of nodes connected by an on-chip interconnect called the network on chip (NoC). Each tile consists of a core, its private L1 instruction and data caches and a slice/bank of shared last level L2 cache. The L2 cache banks in each tile together form the Last Level Cache (LLC) arranged as a NUCA architecture.

Figure 2.2 depicts the micro-architecture of a general-purpose NoC router. For flits coming at the input ports, Virtual Channel (VC) needs to be allocated. After route computation, the flit goes through the VC allocation stage. Each port uses wormhole routing and has multiple VCs per input port. Credit-based flow control is used to avoid buffer overflow and also guarantees free buffer space at the downstream input port. This also controls flit transmission and buffer overflow as well as packet-loss. The buffers in NoC are very simple and mostly FIFO based having say k number of entries. These buffers are managed using head and tail pointers so that the flits do not need to traverse all the buffers before exiting. VCs

are logically grouped under a virtual network (VNet). Packet/flit travels from one router to another over one virtual network to avoid protocol-level deadlock. VC allocator assigns the packet to a VC belonging to the appropriate class (i.e. VNet), similar to the one of the downstream router. Within the VNet the VC allocator looks for a free/available VC to be assigned to the packet. As the flits are output, the VCs become available to be used by the subsequent packets.

The FIFO buffers at the ports of the router consume a significant share of area as well as power of the router. For designing power-efficient routers, we can concentrate on the power consumed by these FIFO buffers. Next, we explore the available memory technologies that can be used to manufacture these FIFO buffers.

2.2 Memory Technologies for Router Buffers

The key characteristics of an ideal memory technology are fast read/write operation, high density, reliability, low energy consumption and cost-effective. However, there is no memory technology which fulfils all of the characteristics. The SRAM memory is fast but at the same time suffers from high leakage power consumption. It is also expensive and has high feature size (i.e. less dense). DRAM memory technology is cheaper and denser than SRAM, but it is slower and unreliable. The solid-state flash memory is reliable and denser than DRAM but has weak write endurance and costly write operation.

Fortunately, router buffers can be designed using different memory technologies that are fit as per the requirement. This enables us to allow the formation of the system that gets the performance benefit from the use of faster technology, energy benefit from the use of power-efficient components and cost advantage by using the cheapest level technology for the hierarchy.

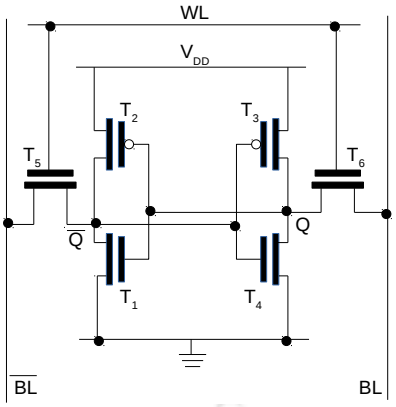


FIGURE 2.3: Schematic view of SRAM cell

2.2.1 Conventional Charge Based Memory Technology

There are NoC router buffers fabricated from SRAM memory technology. Some of the prior studies [48] also exploits DRAM memory technology for the NoC buffers. A detailed explanation of these conventional charge-based memories is given below.

2.2.1.1 Static Random Access Memory (SRAM)

Figure 2.3 shows the schematic view of the SRAM cell. As can be seen from the figure, the core of the SRAM cell contains four transistors, T_1 to T_4, which model two cross-coupled inverters and are used to store the bit information. The stored bit information in these transistors is represented in the form of two stable states, **0** and **1**. These saved states are permanent until the power (Vdd) is applied. The two additional transistors T_5 and T_6 are used for accessing the storage cell during the read and the write operation. The read and write operation for the SRAM cell are described below:

Read Operation: To access the state of a cell, the word access line WL is enabled. This makes the stored stable states available for the read operation in the lines BL and $\overline{BL}$.

Write Operation: To perform a write in the cell, the BL and $\overline{BL}$ are first set to the desired value, and afterwards, the line WL is raised.

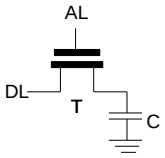


FIGURE 2.4: Representational view of DRAM cell

Other than the design perspectives, there are some other properties (that includes pros and cons) in the SRAM which are important to discuss:

1. The access speed of the SRAM cell is very fast. In particular, as soon as the WL is enabled, the stored stable state is available for the access.
2. The most common SRAM design requires six transistors, thereby it incurs more area on the wafer real-estate and has a lower density than the other memory technologies.
3. The SRAM cell requires a constant voltage supply to retain the data.
4. The SRAM cell is costlier in terms of cost/bit comparison.

2.2.1.2 Dynamic Random Access Memory (DRAM)

Figure 2.4 shows the representational view of the DRAM cell. The DRAM cell is made up of a transistor and a capacitor. The state of the DRAM cell is stored in the capacitor C as a charge. Transistor T is used to access the state. The read and the write operation for the DRAM cell are described below:

Read Operation: To access the state of the DRAM cell, voltage is applied to the access line AL. This makes either the current to flow on the data line DL, based upon the charge stored in the capacitor. In case, if there is no charge, then no current flows to the DL.

Write Operation: To write the cell, the DL is appropriately set to the desired value. Afterwards, the voltage is applied to AL for the extended period, either charging or draining the capacitor.

Because of the more straightforward structure of the DRAM cell, the feature size and the cost of the DRAM is lesser than the SRAM. Despite these advantages,

the capacity of a capacitor to retain the charge in the DRAM cell is shallow, and it requires continuous refresh operation to hold the data correctly. Each refresh operation requires extra time and energy.
The other properties of DRAM can be summarized as follows:

1. The cell structure of DRAM is simpler, and thus, it allows the high density with low cost/bit comparison.
2. The access speed in the DRAM cell is slower compared to SRAM.
3. The refresh operations required to retain the data correctly makes DRAM power hungry.

2.2.2 Emerging Non-Volatile Memory Technology

Nowadays, there are several emerging Non-Volatile Memories (NVM) that are under research. One of state of art [6] lists 12 such NVM technologies: Spin Transfer Torque Random Access Memory (STT-RAM), Phase Change Random Access Memory (PCRAM), Resistive Random Access Memory (ReRAM), Ferroelectric Random Access Memory (FeRAM), Nano Random Access Memory (NRAM), Conductive-Bridging Random Access Memory (CBRAM), Single Electronic Memory (SEM), Polymer, Molecular, Racetrack, Holographic and Probe. From this list, some of the memories have reached advanced development stages and are commercially manufactured, and some are not yet mature.

In this dissertation, we limit our study to the STT-RAM, PCRAM, and ReRAM as it is extensively studied and considered as a viable choice in the interconnect buffers. In particular, we experiment using STT-RAM. These are also backed by commercial industries [49, 50, 51, 52, 53].

2.2.2.1 Spin Transfer Torque Random Access Memory (STT-RAM)

The representational view of the STT-RAM cell is shown in figure 2.5. The STT-RAM cell consists of an access transistor and the Magnetic Tunnel Junction (MTJ) [54] [55]. The MTJ in the STT-RAM cell contains two ferromagnetic layers viz. reference layer and free layer and, the thin insulating metal oxide layer. The metal oxide layer in the MTJ is made up of MgO. The direction of the reference layer is fixed while the direction of the free layer is changed according to the spin-polarized current. The direction of these ferromagnetic layers is used to represent the data bit stored in the MTJ. The anti-parallel magnetization of ferromagnetic layers corresponds to '1' state and the high resistance of the STT-RAM cell. On the other hand, the parallel magnetization direction of ferromagnetic layers represents the low resistance and '0' state of STT-RAM cell.

Read operation in STT-RAM is performed by applying a small voltage difference between the lines (bit-line and source-line) and sensing the current across the memory cell. The generated current is compared with the reference current through sense amplifier, to detect '0' and '1' state stored in a memory cell. Writing on the STT-RAM cell is performed by applying a large positive voltage difference between the source line and bit-line for '0' state and large negative voltage difference for '1' state. The amplitude of current required to alter the direction of two ferromagnetic layers depends on the write pulse duration and, size and aspect ratio of MTJ.

Size of STT-RAM cell is 6-50 F^2 whereas SRAM cell size is 120-200 F^2. As a result, STT-RAM bases buffers will save area of the NoC router. The read energy, as well as read latency of STT-RAM, is comparable to that of SRAM; however, the write latency and energy are relatively high. These can be reduced to a certain extent by relaxing the non-volatility period [56] [57]. The read and write operation for the STT-RAM cell are described below:

Read Operation: To access the state of the STT cell, the access transistor of the cell is enabled and the small voltage difference is established between the lines (source and bit line) (figure 2.5 (b) (2)). This effect causes the current to be

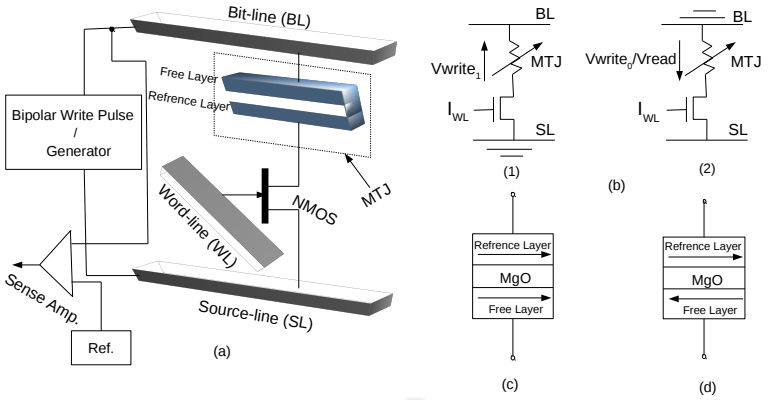


FIGURE 2.5: (a). Conceptual view of STT-RAM cell (b) Schematic STT view with (1) Write '1' operation (2) Write '0' and Read operation (c) Parallel low resistance, representing '0' state (d) Anti-parallel high resistance, representing '1' state

generated across the memory cell, which is compared with the reference current through the sense amplifier.

Write Operation: To write bit '0', a large positive voltage difference is established between the source and bit line (figure 2.5 (b) (2)). To write bit '1', a large negative voltage is established between the lines (figure 2.5 (b) (1)).

The other important characteristics of STT-RAM are summarized below:

1. STT-RAM attempts to accomplish better scalability by utilizing different write mechanism based on spin polarization [58].

2. The STT-RAM cells have more density compared to SRAM but have lesser density compared to DRAM [7].

3. Compared to PCRAM and ReRAM, the endurance of STT-RAM is excellent. However, when employed in the router buffer, the endurance is still considered to be less [8].

4. The write operation of STT-RAM is costlier compared to SRAM and DRAM [7].

Other than the design characteristics, the STT-RAM chip is commercially manufactured and available in the market. For instances, 4Gbit STT-RAM-based perpendicular MTJ at 90nm technology node is fabricated by Toshiba and SkHynix incorporation [59]. Qualcomm and TDK-Headway built the 1Mbit STT at 40nm

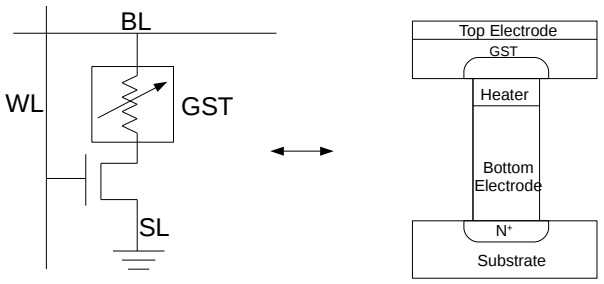


FIGURE 2.6: Representational view of PCRAM cell

technology node [60]. Recently, Intel and Samsung fabricated 7.2M and 8M bit STT at 22 and 28 nm technology nodes [61, 62] respectively.

In this thesis, the terms STT-RAM or STT are used interchangeably.

2.2.2.2 Phase Change Random Access Memory (PCRAM)

Currently, the most mature emerging NVM technology that is under research is the Phase Change Random Access Memory (PCRAM) [63]. Figure 2.6 shows the representational view of PCRAM cell. The PCRAM cell contains phase change or chalcogenide material and an access transistor. The chalcogenide material is generally made up of GST ($Ge_2Sb_2Te_5$, or Germanium, Antimony and Tellurium) and shows two different phases: Amorphous and Crystalline by the application of heat. The high electrical resistivity characterizes amorphous phase and represents the RESET state of the cell. On the other side, the crystalline phase is characterized by low electrical resistivity and represents the SET state of the cell. The read and write operation of the PCRAM cell are described below:

Read Operation: A cell state can be accessed by applying a small voltage across GST. This impact makes the current to be generated as there is a wide resistance gap exist between the amorphous and crystalline stage. The state of the cell is identified by sensing the pass-through current with the help of access transistor and the word-line controlling.

Write Operation: To SET the PCRAM cell, a long duration moderate power pulse is applied that heats the GST above the crystalline temperature and makes

the chalcogenide material crystalline. On the other side, to RESET the PCRAM cell, a high power pulse is applied that heats the GST above the melting temperature and makes the chalcogenide material amorphous.
The other important characteristics of PCRAM are as follows:

1. Due to the significant difference in resistance between the different phases of GST, the PCRAM cell can be used to store the multi-bit information [9].
2. PCRAM is a scalable technology because as the feature density increases, it needs less current for the operations [64].
3. The SET and RESET latency of the PCRAM is larger than the STT-RAM and DRAM [8].
4. The endurance of PCRAM is bound to the limited number of writes. The current write endurance value varies in the range of 10^4 writes to 10^9 writes [65].

The commercial industries focus on PCRAM as a replacement of flash memory technology or to be used as the main memory. Different types of PCRAM chips at different technology nodes are manufactured and fabricated. For example, at the 90nm node, Samsung electronics built the 512 Mb PCRAM chip with 266 Mb/s bandwidth [66]. Later, Samsung fabricated the 8Gb PCRAM chip at 20 nm technology with 40 Mb/s program bandwidth [67].

2.2.2.3 Resistive Random Access Memory (ReRAM)

ReRAM [68] is based on the memristor technology where the resistance change depends upon the polarity, magnitude, and the duration of the applied voltage. Figure 2.7 shows the representational view of the ReRAM cell. The memristor-based ReRAM cell consists of two platinum electrodes with titanium dioxide (TiO_2) metal/oxide interference switches having different oxygen vacancy concentrations. Generally, the metal/oxide interference shows rectifying behaviour with low doping and ohmic behaviour with high doping [69]. In particular, the lower switch of

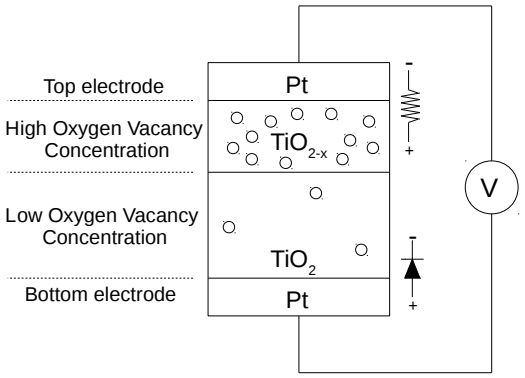


FIGURE 2.7: Representational view of ReRAM cell

perfect titanium dioxide (TiO_2) is electrically insulating, and the upper switch, which has a high having oxygen vacancy concentration (TiO_{2-x}) is conductive. The read and write operation of the ReRAM cell are described below:

Read Operation: To access the state of ReRAM cell, a small voltage is applied across the bit lines. This effect causes the current to be generated that can be sensed to detect the particular state of the cell.

Write Operation: To change the state of the cell, a large voltage is applied across the bit lines. To change the state of the cell to OFF, a negative bias voltage is used, which increases the thickness of TiO_2, which in turn generates the insulating and high resistance ion path. The opposite case is seen in case of positive bias voltage for the ON state.

The other important properties of the ReRAM cell are as follows:

1. The ReRAM memory technology is less mature than the PCRAM and STT-RAM memory technologies [70].

2. In terms of scalability, ReRAM is more efficient compared to STT-RAM and PCRAM. The cell size of 10nm has been achieved, and in future, the cell density of 4-5nm is predicted [10, 71].

3. The write endurance of ReRAM is limited to 10^5 to 10^{11} writes, which is lesser than the STT-RAM and some of the prototypes of PCRAM [71, 8].

The ReRAM technology is still not as mature as other emerging NVMs and is currently under research. Recently, only Fujitsu and Panasonic are jointly working on the second generation ReRAM device [72].

2.3 Challenges to Employ Emerging NVMs in the Router Buffer

The previous section reported the design concepts, essential operations and features of conventional charge-based memory and emerging NVMs. Table 2.1 shows the comparative analysis between the current memory hierarchy technologies: SRAM and DRAM and the emerging NVM technologies: STT-RAM, PCRAM, and ReRAM. To draw the comparison, the following characteristics are used:

1. **Cell Size:** The cell size of the memory cell, measured in terms of feature size (F^2).
2. **Non-Volatile:** This characters is used to explain whether the memory technology is non-volatile or not.
3. **Endurance:** It is the total number of write operations that the memory cell can entertain before it eventually wears out.
4. **Read Latency:** It is the time consumed to perform a read operation in the memory cell.
5. **Write Latency:** It is the time consumed to perform a write operation in the memory cell.
6. **Dynamic Energy:** It is the energy spent during the read/write operation.
7. **Static Energy:** It is the energy consumed during the idleness of memory devices. This energy also includes the energy spent to retain the data correctly.

8. **Maturity:** This characteristic demonstrates that the current memory technology is in the early stage or later stage of development or it is commercially available in the market.

9. **Retention:** It is the ability of a memory cell that how long it can hold the data correctly without performing any refresh operation.

10. **Multi-bit:** It is the ability of a memory cell to keep the multi-bit information.

Features	SRAM	EDRAM	MRAM (STT-RAM)	PRAM	ReRAM
Cell Size F^2	$120 - 200$	$60 - 100$	$6 - 50$	$4 - 12$	$4 - 10$
Non-Volatility	NO	NO	YES	YES	YES
Endurance	10^{16}	10^{16}	$4{\times}10^{12}$	$10^{8} - 10^{9}$	$10^{5} - 10^{11}$
Read Latency	<10 ns	10-60 ns	<10 ns	48 ns	<10 ns
Write Latency	<10 ns	10-60 ns	12.5 ns	40-150 ns	~10 ns
Dynamic Energy	Low	Medium	Low for Read High for Write	Medium for Read Very high for Write	Low for read high for write
Static Energy	High	Medium	Low	Low	Low
Maturity	Product	Product	Advance Development	Advance Development	Early Development
Retention	As long as voltage applied	<<second	>10 yr	>10 yr	>10 yr
Multi bit	1	1	2	>2	>2

TABLE 2.1: Comparative analysis of different memory technologies [6, 7, 8, 9, 10]

As it can be observed from the comparative analysis that the NVM technologies consume considerable write energy and latency, and suffer from weak write endurance. The challenges associated with NVM technologies when employed in the router buffer are illustrated below.

2.3.1 Challenges related to Write Operation

The figure 2.8 reports comparative analysis of average packet latency in SRAM and STT-RAM-based router buffers. To counter the costly write operations, researchers make use of the concept of Hybrid cache and hybrid buffer based router Architecture [1], which is described below. The term hybrid refers to a combination of SRAM and NVM memory technology, where some part of the storage is manufactured using SRAM and remaining using NVM.

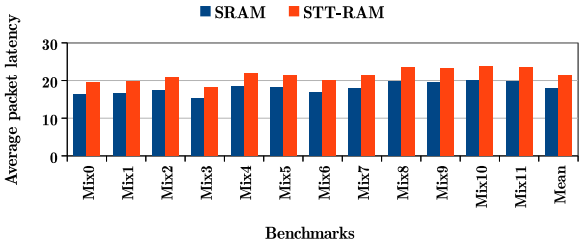


FIGURE 2.8: SRAM vs STT-RAM latency graph for 4×4 mesh network (lower is better).

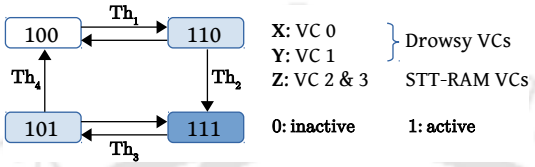


FIGURE 2.9: Hybrid Drowsy VC state diagram

Inside a hybrid structure, the main challenge is to place the right type of blocks in the appropriate partitions. This is done in order to control the number of writes incurred in the NVM portion. In [73], Ahn et al. propose a write intensity prediction technique for the data block placement in a hybrid cache. The prediction decision is made using the relation between the write intensity of the block and the instruction that incurs the cache miss. In [74], the threshold for write intensity is set dynamically based on the application characteristics. They use set sampling to reduce the metadata storage needed for prediction.

There are few research works done on leveraging STT-RAM in on-chip interconnects which reduce leakage power and area of a router. Zhan et al. [1] have replaced conventional SRAM input buffer on-chip with a combination of drowsy SRAM and STT-RAM buffer. They have proposed two hybrid buffer designs, *Hierarchical Buffer* and *Banked Buffer*, to reduce write latency of STT-RAM. *Hierarchical Buffer* composes separate drowsy SRAM and STT-RAM virtual channels. These VCs form multiple levels where the higher level VCs are made of SRAM buffers and are used when traffic is high in the network. This will avoid costly writes to the STT-RAM VCs. The proposal uses one SRAM VC that is always powered-on and another SRAM VC that can be kept in low-power (drowsy) mode when not in use. Additional STT-RAM based VCs are kept power gated in low traffic and

are turned ON during medium to heavy network traffic. Figure 2.9 shows the state transition diagram of the technique. Here, the first level (X) represents the SRAM VC which is ON all the time (represented by 1). Whereas, the second level (Y) represents the drowsy SRAM VC which will be ON and OFF based on traffic load. Lastly, the last level (Z) shows the status of power gated STT-based VC. Initially, when the traffic load is low, only the first level SRAM is active (state 100). When the traffic exceeds a certain threshold (say Th_1), the second level (Y) VC gets activated (state 110). On further increase in the traffic (say at Th_2), the third level STT gets activated (state 111). Note that the traffic is measured here based on the buffer occupancy. In case, when the traffic reduces to medium load from the high load (say at Th_3), the second level (Y) get turned off as the wake-up latency of STT is higher than the drowsy SRAM (state 101). On further reduction in traffic (say at Th_4), the STT-RAM VC gets power gated (state 110). To hide the long latency of write accesses in STT-RAM VCs, the authors propose *Banked Buffer* design. Here, each VC is logically interleaved with SRAM and STT-RAM buffers but physically kept as separate banks. This design allows parallel access to multiple banks which results in improved write latency than the prior design. The authors also discuss how *Banked Buffer* design is beneficial in the case of a traffic hot spot.

Jang et al., use hybrid buffers at the input ports and present a lazy migration technique in [44], to hide the long write latency of STT-RAM. The hybrid design uses a combination of SRAM and STT-RAM buffers at the inputs port of a router. The incoming flits are stored in the SRAM buffer first and then migrated to STT-MRAM to hide STT-MRAM latency which results in increased network throughput. Since every SRAM to STT-MRAM migration may result in more energy consumption due to costly write operation of STT-MRAM buffers, the technique does migration selectively based on traffic condition. To select flits which go through migration, the technique compares network traffic with a predefined threshold value. If the current traffic is more than the threshold, the flit goes through lazy migration, and this helps to reduce the power consumption significantly.

A multi-bank STT-RAM based buffer design is proposed by Jang et.al. [75] for low power On-Chip interconnects using STT-RAM technology. The proposed design provides a large buffer space with very less power consumption. The design uses a virtual channel with multiple banks to hide the multicycle writes of STT-RAM buffers. The incoming flits are alternatively pipelined in the banks for seamless packet transmission. To achieve a reduction in write latency and power consumption, the authors reduce the retention time of STT-RAM design. The design also proposes a cost-efficient dynamic buffer refresh to overcome flit loss due to short retention time. The dynamic buffer refresh technique has very minimum hardware overheads and tries to minimize unnecessary refreshes.

2.3.2 Challenges related to Weak Write Endurance

The write endurance of the STT buffer is defined as the total number of writes that a memory cell can handle before it breaks down. If a memory cell gets written more often than others, it wears out faster than other memory cells. This memory cell will have a lesser lifetime than memory cells with a lesser number of writes. Hence, the lifetime of STT buffers is a major design concern. There are few methods to improve the STT (NVM) lifetime, such as, write reduction technique, wear-levelling technique, and error-recovery methods. This thesis focuses on write reduction and wear-levelling techniques only and does not use any error recovery methods.

In the on-chip network, every VC in a VNet has different numbers of writes. For instance, the control VCs experiences a lesser number of writes as compared to the response VCs. With a different number of writes in the VCs, there is a good possibility that the writes are distributed unevenly inside the VNet, which in turn generates write variation. Similar to cache architectures [76], we classify write variation into two categories:

1. **Intra-VNet Write Variation:** The Intra-VNet write variation is caused due to the uneven write distribution across the VCs within a VNet. In

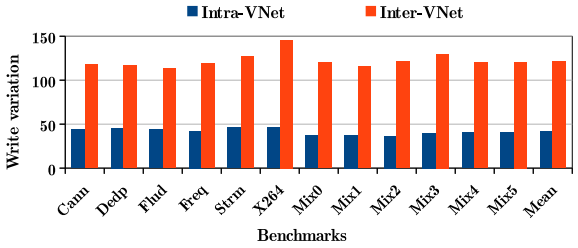


FIGURE 2.10: Write variation in STT-RAM baseline with 16 core, 4×4 mesh network (lower is better).

particular, some of the VCs inside the VNet incur a large number of writes as compared to others. The figure 2.10 shows the percentage of Intra-VNet write variation in control, response and data VNets. As can be seen from the figure, there is a non-uniform write distribution across the VCs. Such uneven distribution of writes results into faster wearing out of heavily written buffers compared to the lightly written ones.

2. **Inter-VNet Write Variation:** The Inter-VNet write variation occurs due to different type and size of packets for different VNets. This generates an uneven distribution of writes across VNets. Figure 2.10 reports the existence of Inter-VNet write variation at the input port. As can be seen from the figure, there is a variable write count among the different input port buffers of a router. Such non-uniform write distribution leads to the early breakage of heavily written VCs as compared to lightly VCs within the same VNet.

To quantify the two write variations: Inter and Intra-VNet mentioned above, we define the coefficients as given in equations 2.1 and 2.3. (i) $Inter\_Var$: measures the average coefficient of write variation across the VNets (ii) $Intra\_Var$: measures the average coefficient of variation inside a VNet. Note that the proposed policies do not require computation of Inter-VNet or Intra-VNet. We use them only to characterize the write-variation present in baseline allocation policy, and we use it for the evaluation of the proposed policies.

Let N be the number of routers, I number of input ports per router, V number of the virtual network at each input port and A number of virtual channel in each virtual network. Also, W_j denotes an average number of writes in all virtual

channel of virtual network j and w_{ja} denotes the number of writes in virtual channel a of VNet j.

We adopt the definition and define the coefficient of *intra-VNet* write variation across the VCs in a given VNet, j, as follows:

$$Intra\_Var_j = \frac{100}{W_j}\sqrt{\sum_{a=1}^{A}\frac{\left(w_{ja} - \sum_{k=1}^{A} w_{jk}/A\right)^2}{A-1}} \tag{2.1}$$

The average variation for each virtual network across all routers for each input port will be defined as:

$$Intra\_Avg_j = \frac{\sum_{n=1}^{N}\sum_{i=1}^{I} Intra\_Var_j}{N * I} \tag{2.2}$$

To calculate the write variation across VNets we can compute the write variation across buffer groups. Inter-buffer group write variation at input port i is defined as follows:

$$Inter\_Var_i = \frac{100}{B_i}\sqrt{\sum_{g=1}^{G}\frac{\left(b_{ig} - \sum_{k=1}^{G} b_{ik}/G\right)^2}{G-1}} \tag{2.3}$$

Here, G is the number of buffer groups at input port, B_i denotes an average number of writes in all buffer groups at input port i and b_{ig} denotes the number of writes in buffer group g at input port i. The average variation for each input port across all routers will be defined as:

$$Inter\_Avg_i = \frac{\sum_{n=1}^{N} Inter\_Var_i}{N} \tag{2.4}$$

Along with the weak write endurance, in the actual execution environment, the lifetime of the NVM memory is further affected by these two write variations, as mentioned above. The lifetime of the VC is defined as follows:

Lifetime: The lifetime of the VC can be defined either as raw lifetime or error-tolerant lifetime [76]. The raw lifetime is determined by the first failure of the VC. Whereas, the error-tolerant lifetime is measured with the raw lifetime and the error recovery methods.

In this dissertation, we have used raw lifetime, which is the basis of an error-tolerant lifetime.

The raw lifetime of a buffer with respect to write variation and write count is determined by two methods:

1. With respect to write counts, the lifetime is defined (eq. 2.5a) as the inverse of the maximum write counts on the virtual channel [77]. To calculate the lifetime improvement of proposed policy w.r.t. the baseline policy, we define lifetime improvement as given in equation 2.5b.

$$L = \frac{1}{\forall_{n=1}^{N} \forall_{i=1}^{I} \forall_{j=1}^{V} \forall_{a=1}^{A} max(w_{j,a})} \tag{2.5a}$$

$$LI = \frac{L_{base}}{L_{pp}} \tag{2.5b}$$

   Here, L_{base} and L_{pp} represent the lifetime for baseline and proposed policies, respectively.

2. We have used above equation to calculate lifetime improvement in proposed policies over baseline. Now, we establish the relation of write variations with approximated lifetime improvement [76]. In terms of write variation, the lifetime improvement is calculated by considering the three factors: (i) the coefficient of Intra-VNet write variation (ii) the coefficient of Inter-VNet write variation and (iii) average write count in a VNet.

$$LI = \frac{W_{avg\_base} * (1 + Inter\_Var_{base} + Intra\_Var_{base})}{W_{avg\_pp} * (1 + Inter\_Var_{pp} + Intra\_Var_{pp})} - 1 \quad (2.6)$$

In the above equation, the W_{avg_base} and W_{avg_pp} are the average write count in baseline policy and proposed policy respectively.

2.3.2.1 Improving the lifetime and the endurance of Non-Volatile buffer

Another challenge with the employment of NVM as an on-chip router buffer is the weak write endurance. In a real-time application, low endurance affects the lifetime of the NVM buffers. Additionally, the lifetime of an NVM buffer is affected by unwanted system-generated write variation (categorized as an Intra and Inter VNet write variation as reported in section 2.3.2). To improve the lifetime and to mitigate the unwanted write variation, there have been several research contributions [76, 73, 78, 79, 80] related to wear-levelling in STT-RAM based caches. Not much work is done in this domain for NVM based NoC router buffers.

Here we discuss some techniques used for intra-set wear levelling in caches. In i2wap [76], Wang et al. proposed an intra-set wear levelling technique to increase the lifetime of NVM based cache. The policy invalidates a cache block after a fixed number of writes determined by the Flush Threshold (FT). A counter is associated with cache bank, and this counter is incremented after every write to the bank. A probabilistic method is used to select a block for invalidation rather than a deterministic approach. The replacement policy chooses a write-intensive data (hot data) inside the cache set as a victim. In case this is a hot-block then on its access it will get placed in a different location in the cache set, which will perform the wear levelling. Another technique, EqualChance [77] by Mittal et al., uses a counter with every cache set. The counter is incremented with each write to the cache set. If the counter reaches a threshold, the block is transferred to an invalid cache line in the cache set, and the counter is reset. In LastingNVCache [78] presented by Mittal et al. associates a 4-bit counter with each cache block to maintain the number of writes received by the block in a single

generation. If the counter reaches a threshold, the write operation is left, and the block is invalidated without updating the replacement information. In another technique, WriteSmoothing [79], the cache is partitioned into multiple modules of the equal number of cache sets. Here, the write variation inside the module is reduced by transferring the data from hot sub-ways to cold sub-ways within the module. In ENVLIVE [80], additional small storage, called HotStore, made of SRAM is used to store the write-intensive cache blocks. Only the blocks with more number of writes will be eligible for the placement in the HotStore.

Some contributions related to inter-set wear levelling for caches appear in [81, 82]. A grouped access intra-set swapping is proposed in [81]. The method changes the cache set mapping between the heavily and lightly written set of the group with the help of the counters. This way the heavily written set will incur lesser writes in the next interval and the lightly written set in the current interval will take a load of more writes in the next interval, resulting in wear levelling across cache sets. In [82], Soltani et al. proposed to partitions the cache into multiple clusters. The mapping for the clusters is changed during the execution using the write intensity, mapping history, and the number of clean/invalid blocks. The mapping ensures that the writes get distributed evenly across clusters resulting in inter-set wear levelling.

In the context of NoC buffers, [83] proposes the use of hybrid buffers. As STT-RAM is denser than SRAM, the paper uses four times the number of buffers than SRAM while replacing certain SRMA buffers. This gives more number of buffers that are used in a pipelined manner to overcome the writing latency. To mitigate the write latency further, the work uses conservative programming conditions and at the cost of reduced retention time and reduced non-volatility. This, however, leads to unreliable device behaviour. To take care of these stochastic switching effects of STT-RAM an efficient error control and coding scheme is proposed.

2.4 Router power saving approaches

This thesis proposes to reduce the energy consumption of NoC routers by employing pure and/or hybrid memory based buffers at the IO ports. With a similar aim of energy efficiency, several researchers have contributed by proposing novel design architectures of the NoC routers. These include proposals for bypassing the router pipeline, scaling the frequency of the routers, using multiple planes of NoC links, power gating the routers when not in use, etc. The decisions to use the alternatives are mainly based on the dynamic network traffic conditions. This section explores all such existing approaches.

2.4.1 Pipeline Bypassing

Boyapati *et. al.* [84] proposed a distributed power gating method by modifying basic router architecture. The router has basic components, and Fly-Over (FLOV) link to bypass packets when attached cores to the router are powered down by OS. The incoming packets to these routers do not go through router pipeline stages; rather, it travels through FLOV link without making any turn. A Hand-Shake Control logic (HSC) block is introduced, connecting to all the neighbouring routers, which implements the handshake protocol between adjacent routers required before power-gating a router. Two sets of Power State Registers (PSRs) hold the power states of the immediate neighbour routers and the nearest powered-on routers (logical neighbours) in each direction. When a core goes down, the attached router waits for the packets coming/going from/to the core for a certain number of cycles. If there are no packets, the router sends a signal to its neighbours indicating its *Draining* state. During this state, no neighbouring routers can transmit new packets but can complete current packet delivery. Once all the buffers are empty, the router informs its neighbours about its power gating state and neighbouring routers establish a connection via FLOV link of a power-gated router using Hand Shake Controller (HSC) and associated protocol. A power

gated router becomes active when its core becomes active, or a packet comes to the router for the attached core.

Hossein *et. al.* presented a pivot based power-gating technique, Sponge [85], to save leakage power consumption of routers. A predefined column of routers (called pivot) in the centre of the mesh network is kept powered-on. The routers are architecturally modified and have bypass links which are used for packet traversal when the router is in the power-off state. These power gated routers are not woken-up for travelling straight. Any injected or ejected packets, and turning packets are forced to take a turn at the boundary pivot column. The routers of the adjacent column of the pivot are kept shutdown unless traffic exceeds the predefined threshold. When there is high traffic in network, the adjacent column routers gets activated, increasing the column width of the pivot and are kept active for a predefined number of cycles. In Sponge, even if a single router exceeds threshold traffic, an entire column of routers get activated. Whereas, to shut down, a column of routers all the routers in the column should have traffic below the threshold. Similar to FLOV, a flit takes a turn at the active router or at pivot in the worst case.

Ejaz *et. al.* proposed a Dual Data Rate (DDR) NoC router architecture composed of router's DDR data path to increase network throughput [86]. DDRNoC utilizes the benefit of the faster data path in the router and allows two flits to share the same data path in a cycle for both clock edges. In a router, DDR mode is enabled for a data path stage, switch traversal and link traversal, when more than one packets are competing for it. Also, the data path operates on DDR mode for two flits belonging to the same packet. In all other cases, the data path operates in single data rate. Despite the improved network throughput, DDRNoC has high packet latency. DDRNoC supports about 25% higher throughput than Short-Path [87] but has about 50% higher packet latency To solve the above issue, the authors proposed a pipeline bypassing method, Freeway-NoC [88], which uses DDR data path and improves performance caused by data path delays. FreewayNoC restricts pipeline bypassing to the flits which only go straight in the router to

preserve throughput. These flits directly traverse the switch and link without undergoing allocation stage if the destined output port is free. Freeway-NoC exploits look-ahead signalling mechanism which uses a virtual channel selection mechanism at the output ports instead of VC allocation to accommodate pipeline bypassing and check on next-route computation. Unlike DDR-NoC, FreewayNoC does not add any complexity to routers data path preserving the DDR packet rate, while reducing packet latency.

Muffin [89] proposed by Farrokhbakt *et al.* is a coarse-grained power gating scheme that aims to minimize the power and delay overhead of power gated router. Here, by using some extra buffers and employing the bypassing mechanism, the power gating period increases. In order to facilitate this process, an extra five flits sized buffer, a small logic, and a light-weight power gating controller are added. To do that, two types of the buffers *Bypass* and *interject*. Based on the type: straight, inject, eject and turn off the packet routes and using multiplexing logic, different buffer(s) are employed at each port of the router without needing to power ON the router. Additionally, a priority mechanism is also employed to schedule the shared buffers and multiplexing logic for each type of route packets. High priority is given to the straight route packet than the turn and inject packet.

TooT [90] proposed by Farrokhbakht *et al.* is a scalable and effective pipeline bypassing method which reduces the latency overhead and energy consumption of power gating techniques. TooT is based on the observation that mostly a router is woken-up due to the arrival of a straight packet (packets which do not take a turn from the router). TooT aims to minimize the number of wake-ups by providing a bypass path for straight packets. The routers are augmented with a bypass path and one-flit input buffers to facilitate pipeline bypassing for straight packets. To implement this, it uses free-to-forward signals which are used to communicate with neighbours on the state of the bypass latch.

Power gating routers saves power, but consume wake-up latency that may hamper performance. To mitigate this wake-up latency, in certain cases a bypass path may be useful. A technique that decouples power gating with bypass is proposed

in NoRD [91]. The bypass is achieved by forming a unidirectional ring in the network by connecting the input and output ports of all nodes through two-level coordination. For packets injected and ejected via local ports, there is additional data-path connecting to the bypass ring. NoRD ensures deadlock and livelock free adaptive routing and saves leakage energy.

Another technique has pipeline bypassing. However, the objective here is not to use bypass route when the router is power gated, but to use bypassing depending on the traffic rate in order to optimize the packet latency. The design ShortPath [87] by Psarras *et. al.*, proposes a non-speculative dynamic pipeline-stage bypassing technique to reduce flit latency in the network. The design guarantees that each flit will spend the minimum number of cycles in each router for low or high network load. When a head flit arrives both the stages of VC allocation is performed in parallel with the help of VC Request Queue (VCRQ) introduced in ShortPath. Further, the switch allocation (SA) stage is performed, and the request is enqueued in the SA Request Queue. ShortPath allows fine-grained pipeline partitioning of the allocation tasks and enables an always-productive pipeline-stage bypassing. This dynamic pipeline-stage bypassing technique is collapsible; hence a flit may traverse a single, two or three-stage pipeline in the router. The worst case pipeline depth, traversed by any flit, is all the three-stages of the router.

Some recent methods allow packets to bypass not a single router but across multiple routers. Krishna *et.al.* presented a Single-cycle Multihop Asynchronous Repeated Traversal, SMART [92], NoC design. They propose to embed repeaters within each router's crossbar so that the signals can travel up to multiple hops. In SMART's flow-control mechanism, an arbitrary multi-hop path is set and traversed in a single cycle. The flits are buffered in the start router, and the single-cycle multi-hop path begins from this router. The Switch Allocation (SA) in SMART router happens in two stages, switch allocation Local (SA-L) and switch allocation Global (SA-G). An SA-L is selected for each output port from its buffered (local) flits. Once the winner is chosen, in the next cycle, the output port winners broadcast a Smart-hop setup request (SSR) up to a predefined number (k) of hops from that output port. In the SA-G stage, all intermediate routers arbitrate among the

received SSRs. This guarantees that only one flit that wins both SA-L and SA-G gets access to the output port. In the next cycle, a single-cycle multi-hop bypass path gets set up (from current router to next k^{th} hop), and the flit gets latched into the k^{th} hop. In the case of multiple SSRs, a priority mechanism based on locality or bypass is applied to avoid SSR conflict. In the locality based approach, local flits get priority over bypass flits. In bypass based approach, bypass flits get priority, where the flits from the furthest router are considered to have more priority over flits from nearer routers.

SMART++ [93] is another multi-hop traversal design proposed by Perez *et al.*. It combines SMART bypass [94], multi-packet buffers, Non-Empty Buffer Bypass (NEBB) [95] and per-packet allocation using grant-and-hold circuits. The technique relies on a more aggressive VC reallocation policy and supports the bypass of buffers even when they are not completely free. SMART bypass [94] supports flits to cross multiple routers in a single cycle. Whereas, SMART++ supports multi-hop bypass irrespective of inter-router buffer empty/non-empty status. SMART++ introduces three improvements over SMART: multi-packet buffers, Non-Empty Buffer Bypass, and packet by packet arbitration. It allows more than one packet to be held in a router buffer (with a buffer size of the largest packet in the network). NEBB [95] is a bypass policy that utilizes the bypass by removing empty buffer limitation. The NEBB allows bypassing a non-empty buffer without advancing packet to the output port.

Bypass paths can also be used to increase performance, reliability and low-cost fault-tolerant design. A pipeline bypass mechanism for a 2-stage router is proposed by Wang *et al.* [96] to achieve faulty tolerance. To provide a fault-tolerant route computation, look-ahead routing with an extra stage in a downstream router, to compute output port, is used. For fault-tolerant Virtual Channel Allocation, a register is added as a bypass path for input VC arbiter. When the arbiter is faulty, the output VC in the register is selected as the default winner. In the 2-stage router, there are two parallel switch allocators, non-speculative and speculative. If the switch allocator handling non-speculative requests is faulty, the arbiter is selected from another switch allocator to process the non-speculative requests.

In the next cycle, the flits enter in speculative switch allocation stage and get processed successfully. If the crossbar is faulty, no flit will be transferred to the downstream router. To solve the fault issue in the crossbar, Wang *et al.* proposes to use two bypass buses in the router for flits to cross the faulty crossbar. A bypass bus is used for flit traversal in x-dimension or the local router, and it connects all output ports. The other bypass bus is used to connect all the output ports in the y-dimension. In most cases, there are no conflicts between these two bypass buses and the flits can be processed parallel under a heavy workload.

Noghondar *et al* proposes a priority-based mechanism to reduce latency using priority in bypass-based NoC. The input ports have predefined priorities associated with them, and the arbiter allocates output port based on the input port priorities. The priorities of the input port can be decided dynamically or statically. In the dynamic approach, the priority is based on the port injection rate in each cycle, and the router determines the injection rate and priority. Whereas in the static approach, the priorities are fixed. The priority-based allocation may result in a starvation problem. To solve this, there is a counter associated with each input port which is incremented. Here, after a fixed number of cycles and depending on the value of the counter, an input port gets access to an output port regardless of its priority. The bypass flits traverse routers through the bypass path so that the number of hops is decreased. Hence, by using a bypass path with priority, the number of hops counts can be decreased significantly.

Similar to these, other research contributions, based on router pipeline bypassing are available in the literature [97, 98, 99]. All the works mentioned above show significant improvement in power saving. However, it is to be noted that pipeline bypassing is effective during low traffic loads, where simple and deterministic routing algorithms are sufficient.

2.4.2 Frequency Scaling techniques

The voltage-frequency scaling approach increases or decreases the frequency of the router or its components based on the high or low traffic at the router. The traffic is calculated using buffer occupancy or link utilization. This kind of approach is mainly done to reduce static power consumption, dynamic power consumption or both.

To implement voltage scaling, we can have multiple resources in the network and scale control their individual voltage/frequencies. A proposal dark-NOC [100] by Bokhari et al., uses an NoC architecture, having multiple layers of identical and physically different routers. Each layer is optimized for a particular voltage-frequency range. At a time, only one layer is active, and other layers are kept dark. The authors have also given architectural support for seamless integration of multiple network layers and a fast inter-layer switching mechanism without dropping in-network packets.

Another proposal scales the voltage of the links instead of the complete router. Shang et al. [101] proposed a history-based dynamic voltage scaling with links which is a significant consumer of power when bandwidth increases. The proposed idea reduces link dynamic energy consumption under low link utilization using link frequency and voltage scaling. The frequency and voltage of the links are adjusted based on past utilization. Soteriou et al. [102] proposed a self-regulating power-aware interconnection network that turns the links on or off in response to high and low traffic in a distributed fashion. The network load is distributed using adaptive routing where the proposal adapts dynamically to variations in network traffic. It was noted in [103] that dynamic link voltage scaling gives better performance when the demand on the network grows. The paper observed that the network with all links set to the same voltage level with adaptive routing to distribute the load, outperforms the network having links that dynamically scale the voltage to reduce power consumption. Adaptive routing responds more quickly to network demand changes. However, adaptive routing with static link voltages also outperforms, in many cases, when adaptive routing is used with dynamic link voltages method.

Voltage frequency scaling at the VC-level is proposed in [104] by Matsutani et al., that results in reducing the switching power of on-chip networks keeping the leakage power consumption small. The proposal uses runtime power gating to individual virtual channels to reduce leakage power.

The work [105] reduces link dynamic energy consumption under low link utilization using link frequency and voltage scaling. The frequency and voltage of the links are dynamically adjusted based on past utilization. The idea is augmented with fly-over and decouple the router frequency from the link frequency. This enables packets to travel via wires only and get buffered only when necessary.

The decision to scale the router frequency can be based on a variety of metrics. The main one being the network traffic. It would good to know the traffic load beforehand so that the frequency scaling can be done in advance. In [106], Hesse et al., utilize the highly predictable properties of cache-coherence communication to derive more specific and reliable NoC traffic predictions. The Voltage and frequency level of upstream and current routers are adjusted based on the prediction. The approach employs application characteristics and not purely network-related metrics. The application behaviour is used to predict DVFS relevant properties of NoC traffic. This results in a reduction of voltage/frequency mispredictions and leads to more power-efficient NoC implementation without reducing application throughput.

Routers running at different voltage and frequencies are useful for NoC in the dark silicon setting. Each router is designed to run at a predefined VF setting. Malleable NoC [107], is such a proposal. Based on the number of memory controllers, the cores on the chip are logically divided, and memory islands are formed. The applications are allocated to each memory island in round-robin order and mapped based on the L1 miss per kilo instructions (MPKI) and the distance of the core from the memory controller. Each island can be operated at a different VF setting make optimal energy consumption.

Similar to these, other research contributions, based on router frequency and voltage scaling are available in the literature [108, 109, 110, 111, 112, 113, 114, 115, 116, 117].

2.4.3 Power Gating

Soteriou *et al.* propose a component-based power gating technique that power gates components such as ports and links [102]. The power gating of these components is done in response to network traffic condition.

In [118], Matsutani proposed a technique where the power supply of each router components can be controlled individually. The components which are not transferring packets are power gated which results in a significant amount of leakage power reduction. However, power gating each component increases communication latency since a significant number of cycles are needed to wake up any component when a packet arrives. This results in performance degradation. In order to enhance performance, they proposed early wake-up methods where a packet arrival is detected, and the corresponding component is activated. In the first method, the current router uses two hops look-ahead routing and activates the next hop immediately after the route computation is done. This can save one router pipeline cycles length worth of wake-up latency. The proposal uses a separate wake-up control network to send the early wake-up signals. In order to control the first hop wake-up latency, network interface triggers the activation of first and second hop routers.

Another non-blocking power gating scheme, Power Punch [119], proposed by Chen *et al.* allows obscuring the costly power gated router wake-up latency by sending the wake-up signal three-hops in-advance. However, to facilitate this process, a separate wire is needed between every router pair (that is three hops away) for sending the wake-up signals. This overhead can be minimized by incorporating encoding of wires for the wake-up signals.

In the dark silicon scenario, network power gating is done by fine-grained topological sprinting by Zhan *et.al.* [120]. In core-sprinting, when a burst of computation arrives, a number of cores get activated and run for a short period of time. In irregular topological sprinting, there is only one core kept active, called the master core. The placement of the master may depend on the delay in transmission latency for thread migration. The core which runs the OS (since it is always activated) or the core next to the memory controller can be a good choice for the applications with intensive memory accesses. For efficient core-sprinting, an equally efficient low power sprinting interconnect is needed. In [120], authors propose a flexible sprint policy which activates a subset of network components to connect a certain number of cores for different workloads. A deadlock-free routing along with short communication delay between master and other nodes is required to run an efficient core-sprinting. The topological sprinting activates a subset of routers, and the rest of the network is kept in the power-gate state to save leakage power consumption.

Samih *et.al.* propose an interconnect power management approach called Router Parking to save leakage power in on-chip network [3, 121]. The main idea is to power-gate (park) routers attached to dark/sleeping cores. However, power gating such routers may lead to the partitioning of the network, and this can lead to network discontinuity. To solve this issue, Router Parking has three methods: Aggressive Router Parking, Conservative Router Parking and Adaptive Router Parking, which maintain network connectivity and deals with the trade-off between power and performance. (i) In Aggressive router parking, all the routers attached to sleeping cores are kept power-gated. It uses a strongly connected component algorithm discussed in [122] to identify the number of connected components in the network. Once the partition is detected, a minimum number of routers are activated to connect the network. This approach may lead to the detour of packets which results in unexpected packet latency during high network load. (ii) The Conservative router parking method parks fewer routers to minimize the impact of packet latency. Here, a router will be parked only if none of its direct or indirect routers are active. (iii) The adaptive router parking approach selects between

aggressive, conservative or no router parking based on run time utilization of the interconnect. The policy selects two thresholds Min and Max. If the network utilization is less than Min then aggressive router parking gives maximum savings. If the utilization is more than Max there is no router parking selected and all routers are kept active. When the utilization is between these two thresholds, the selection of conservative router parking gives better latency performance.

Recent works done on router power gating methods show that the idleness of the routers is distributed across its components and that the routers as a whole do not provide enough idle time for efficient power-gating [123]. Therefore power gating at the more finer granularity, such as virtual channels (VC), are more relevant candidates since they can enjoy continuous idle periods and low utilization [124, 104, 125].

Mirhosseini *et.al.* [126] propose an energy-efficient virtual channel power gating mechanism which dynamically adjusts the number of available virtual channels depending on the network traffic, thus achieving maximum leakage energy savings without affecting the performance during high traffic load. Unlike [104], this policy uses the ratio of the number of wins to the number of loses of VC allocation requests at each port, as a power gate metric for the VCs. To implement this, it uses two counters: one to record the wins and second to record the loses, at each out port. To decide whether to power-gate or to power-on a VC, the counter values are checked against the thresholds. The routers are divided into three categories, hot, warm and cold, based on link utilization. Each category of the router has a different pair of threshold to power-gate and power-on the VCs. Hence, the cold routers have a smaller threshold, and hot routers have large threshold values.

Partial power gating scheme for a bidirectional network is proposed by Wang *et.al.* [127]. The design uses direction-slicing that divides the router into two slices based on the directions in the topology (input or output direction) where each slice has a unidirectional channel in every dimension. For example, in a 2D mesh network, all $East\_Out$ channels of even rows, $West\_In$ channels of odd rows, $South\_In$ channels of even columns, $North\_Out$ channels of odd columns,

and Local channels are always kept ON to form the ever-on subnet. The other remaining channels, belonging to gated slice are power gated to save leakage power consumption. To utilize the direction-sliced power-gating, the router is divided into two power zones across channels, and corresponding VC buffers, crossbars, and output latches. Due to the circular dependency in the ever-on sub-network, there is a chance of deadlock occurrence for which the design introduces a deadlock recovery scheme. The partially ON routers forbid access to certain ports; hence the design introduces improved routing algorithm which handles both the cases: (i) routing packets on the ever-on subnet when the gated slice is powered-off and (ii) utilizing all ports when the sliced router is fully on.

Liam *et.al.* proposed a new router architecture, DimRouter [128], which supports three modes: *normal*, *dim* and *dark*. A *normal* mode router has all the components in active state; whereas the dark router is fully power-gated and non of its components are active. The *dim* routers are kept into a near-zero power state by deactivating majority of its power-consuming components and maintains necessary connectivity in the network. The router modes are selected based on the status of the associated core/processing element. If a router is attached to an active core, the router will be kept in *active* mode. If the attached core to the router is in an inactive state, the router goes to *dark* mode. If some routers in *normal* modes are disconnected, then a minimum number of routers are kept in *dim* mode to maintain connectivity in the on-chip network.

For power gated routers, if there are alternatives paths available, then it does not hamper network performance. Alternate bypass paths can be established using various design alternatives. One such method is to have additional virtual channels. Wang *et.al.* propose an Express Virtual Channel-based (EVC-based) [129] power gating technique. In this technique, there are predefined multiple virtual bypass paths between source and sink routers in each direction. The sink router (of the virtual bypass path) has additional buffers to hold incoming packets on the path. The packets can take these virtual bypass paths to bypass power-gated or power-on routers. The technique modifies the router structure and the modified router has a power-gate control unit, handshake control unit and EVC control

units at the sink and source routers. This supports (i) transmission of packets via normal path even is those routers are power gated, (ii) This also enables packets to avoid powered-off routers even if it does not take a virtual path. To bypass the routers, it is necessary for packets to take EVC path; hence the routers always try to send packets via virtual bypass path if available. This reduces power consumption even with high network traffic.

Another approach to bypass packets is to use bypass path when the router is power gated and in case of the traffic increases, the router can be powered-on. Zheng *et.al.* propose such energy and performance efficient power-gating router architecture, EZ-Pass [130], which hides large wake-up latency with significant leakage energy saving. The EZ-Pass router consists of an EZ-Pass switch (i.e. bypass route) to transmit packets under medium to low traffic load without the packets going through all the pipeline stages as in the conventional router. For the power gated router, the policy uses the EZ-Pass route to send the first flit. If there are more incoming flits during the processing of the first flit, then router gets activated.

The technique discussed in [131] uses a Duty Buffer based fine-grained power-gating scheme to reduce the leakage power consumption of the VCs in the routers. While the VCs are power gated, any incoming flit is written in the duty buffer, and the process of waking up the VC is started. The flit order in the buffer is maintained to keep the transmission order correct. Also, to avoid any protocol level deadlock, the order of packets are maintained across all packet classes (i.e. VNets). The usage of duty buffer saves leakage power consumption that would be consumed by the powered-on VCs during low network load. It also reduces latency caused by VC wake-up upon flit arrival.

Most of the NoC power gating methods rely heavily on network traffic. In order to benefit from power gating, an adequate idle period (namely, "break-even time") of routers should be guaranteed to make sure they are not frequently woken up and gated off. Also, most of these techniques do not account for the underlying core/application status and might result in sub-optimal power gating decisions as

these are based on only network traffic. Therefore such methods achieve power reduction at the price of some (often considerable) performance loss.

2.4.3.1 Power-Gating Vs NVM

There are a few NoC based techniques proposed which use NVM based router buffers[1]. In [1], authors compared proposed policies, hierarchical buffer (HB) and banked buffer (BB), with All_SRAM_PG, All_SRAM_PG_LA [132] and NoRD [91] with baseline, All_SRAM, design. All_SRAM_PG and All_SRAM_PG_LA use pure SRAM based buffers and power gates it when not in use. All_SRAM_PG power gates all buffers incur 5.82% of energy overhead due to frequent router wake-up; whereas, All_SRAM_PG_LA and NoRD show energy reduction of 11.9% and 16.4%, respectively. The proposed policies use hybrid buffers (Drowsy-SRAM and STT-RAM) and activate STT-RAM VCs at high network load. This achieves 31% and 26.3% of total energy savings in HB and BB policies over baseline design.

There are a few research done on NVM based caches to reduce the energy consumption of conventional cache design. In [133] shows the leakage energy can be as high as 80% of total energy consumption for an L2 cache in 130nm process. Thus, using STT-RAM last-level caches can effectively reduce standby leakage power in caches. In [134], the proposed strategy first calculate Average Memory Access Time (AMAT) of running applications on the CMP for each predetermined interval by collected systems memory traffic. Based on the monitored AMATs, the Hybrid distributed LLC is reconfigured by selecting the SRAM or STT-RAM bank at the runtime. The policy reduces 39% of total energy consumption over SRAM baseline design. The proposed policy Obstruction Aware Policy (OAP) [135] identifies the processes that obstructed the other running processes in the multi-core system and bypass their data from the NVM based L3 cache upon a hit or miss except for read hit. It reduces energy consumption by 64%. In DASCA [136], the write blocks are classified into three categories: dead-on-arrival fill, dead-value fill, and closing-write. It identifies such blocks if the block falls under dead-on-arrival

[1]The techniques are discussed in section 2.3

fill or dead-value fill category it is sent from upper-level cache to main memory. If the block falls under the closing-writes category, it is sent from main memory to upper-level cache. This leads to an energy saving of 47% over SRAM baseline design.

All these existing proposals show that replacing SRAM with NVM is beneficial in energy savings.

2.4.4 MultiPlane NoC

Virtual channels are flow control technique for on-chip interconnection networks which avoid deadlock and improve link utilization and network throughput. However, in the resource-constrained multi-processor system-on-chip (MPSoC) space it is still questionable, because of their huge overhead in terms of area, power and cycle time degradation. An alternative is the Multiple Physical (MP) network which we discuss next. In a MultiPlane-NoC the wires and buffers are partitioned across several sub-networks, and each node is connected to corresponding routers in all the sub-networks.

Catnap presented by Das et al. proposed a scheme to choose the sub-network (subnets) from the multiple narrow networks for power gating [137]. The subnets are given levels of priority. New packets are injected in higher priority subnets only if the currently active subnets are getting congested. This ensures longer durations when higher-order subnets are powered off saving on energy. The technique makes sure the maximum power gating along with the connectivity in the network without degrading the performance in case of high traffic. It maximizes the sleep cycles while reducing the number of switches between the power states of the subnets. Here, the subnets and its routers are powered off when it has empty input buffers, and it has been idle for a predefined number of cycles, and the network is currently not congested. If any of these conditions are violated, then the subnet is powered-on. Note that congestion is detected by using regional/local traffic information.

In [138] Yoon et al. present a comparative analysis of Virtual channels (VC) and multiple physical (MP) network alternatives. Both the approaches give better performance, quality-of-service, and deadlock-free packet-switched service in network-on-chip design. The contention in the network resolved dynamically, VCs provide lower zero-load packet latency than MPs; Whereas MPs can be designed using simpler routers and narrower channels which decreases power dissipation and area occupation. The authors discuss the strength and weaknesses of each alternative designs with both FPGAs and standard-cell libraries and system-level simulations.

A multimode on-chip interconnect called SuperNet [139] is proposed by Bokhari et al. The design is configured to run in three different modes: energy efficient mode; performance mode; and, reliability mode. The proposed design uses two parallel multi-vt [100] optimized packets switched NoC mesh. Energy mode saves more power, whereas performance mode enhances core IPC on high MPKI applications. The reliability mode provides protection against soft errors in the routers data path. In the multi-NoC placement of the traffic distributed or demultiplexer is important. In [140] Yadav et al. propose network demultiplexer placement at the control plane of switch allocator of the router. The placement improves static power efficiency as compared to conventional data placement design. Another similar work [141] places the demux at the routing logic.

In [142], Gilabert et al. proposed a simple and efficient approach to VC implementation to save more area and power. Having multiple VC involves having more number of buffers in the router and then transmitting them over a shared physical link. Instead of this, the paper proposes to replicate the entire switch rather than replicate only the buffering resources for each physical link. This makes the design more area and power-efficient while running on higher speeds.

Wu et al. proposed a hybrid Multi-NoC design, HM-Mesh [143]. It combines a hybrid CMesh and Mesh architecture and leverages CMesh network for power efficiency at low network utilization. As per the network load, the paper proposes a power gating and packet scheduling strategy. At low traffic, it uses only the CMesh,

and as traffic increases, the other network is turned on. HM-Mesh schedules packets to different subnets according to the traffic load dynamically.

Multi-NoC can be composed using different types of networks. Flores et al. propose a heterogeneous interconnect design [144] for CMPs, which consist of two types of sub-networks: low-latency wires for critical messages and low-energy wires for non-critical messages. Firstly they propose a technique called ReplyForwarding which categorizes the messages into these two types. The reply message carrying data is partitioned into a short critical message containing a sub-block of the cache line requested by the core and the whole cache line as a non-critical message. These two parts are sent over different interconnects to achieve energy-efficient transmission. Another heterogeneous interconnect is presented by Abousamra et al. [145], where they use a circuit-switched and another packet-switched network. Control packets being critical are sent via packet-switched subnet, and data packets are sent via the circuit-switched network. In order to save energy, the voltage and frequency of the circuit-switched network are reduced using an algorithm that also performs circuit reservations for latency reduction. Spatial Division Circuit Switched NoC [146] based on Spatial Division Multiplexing (SDM) is presented by Ejaz et.al. The paper claims that the circuit-switched (CS) network can guarantee throughput with the low area and power budget. They develop a network evaluation platform to study the performance of SDM over CS. The work shows that under uniform traffic, with requests of uniform random bandwidth (BW) requirement, a less flexible network outperforms a network with higher flexibility due to the 'stray requests' phenomena.

2.4.5 Power saving at Buffer level

We have noted that in NoC a significant consumer of the energy is the buffers at the ports within the router. These buffers are used to form the VCs and sets of VCs for VNets. Researchers attempt to reduce the power consumed by the buffers by either replacing them with emerging memory technologies or by power gating them. The main aspect to consider is the asymmetric read-write timing of

these emerging memories. Kline et al. proposes [147] a control mechanism based on *shift-register* nature of spintronic domain-wall memory (DWM) to replace the conventional memory buffers for the NoC. The design ensures that reads and writes can be aligned more effectively for simultaneous access in the same cycle.

Area and power consumption can be reduced by reducing the number of physical buffers at the ports and assigning them dynamically to different VCs. Flich et al. proposes such an NoC router architecture, CUTBUF [148], which dynamically assigns virtual channels to Virtual Networks (VNet) based on the traffic load of the VNet. It reduces the number of physical buffers in routers which saves area and power consumption without compromising NoC performance. The CUTBUF architecture is designed to work deadlock-free at network and protocol levels. The design reuses the same buffer for different traffic loads, and the VCs are not statically assigned to any VNet. However, a minimum number of queue imposed by the coherence protocol are implemented in the design.

We can also save power by having as many physical buffers at the ports, but power gating them when not in use. In [2], Zoni *et al.* proposed a power-gating method BlackOut, which individually controls the power supply of NoC buffers. The virtual channel buffers are dynamically switched on and off based on the traffic condition. BlackOut uses two methods: flow balancing and late binding to control input buffer power gating. Flow balancing uses the output port state and the active buffer state to increase or decrease the number of active buffers at the downstream router. Whereas, late binding approach optimizes the allocation of a packet to a buffer using REMAP(ing) at an input port of a downstream router. The REMAP function dynamically binds the physical buffers to different virtual channels. Another work FlexiBuffer [149] proposed by Kim et al., employs a fine-grained power gating method. Here the buffers are either in sleep or active state. Initially, a certain number of buffers are active in a router, and the other buffer gets activated when congestion is detected in the network.

Similar to these, other research contributions, based on router buffer optimization to save leakage energy are available in the literature [150, 151, 152, 153, 154, 155,

156, 157, 158, 159, 160, 161]. Certain other research approaches take the bufferless router approach [162, 163, 164, 165, 166]. However in high traffic load these may result in performance degradation.

These approaches were mainly proposed for SRAM based router buffers which are not suitable for pure NVM or NVM based hybrid design due to the challenges related to weaker write endurance and costly write operation of NVM. Thus, the thesis proposes different longevity approaches to overcome the NVM challenges.

2.5 Summary

With a large number of cores integrated on-chip, the basic building blocks of modern computing systems (CMPs) requires an efficient switch-based Network on Chip (NoC). Design space of NoC is tightly constrained as the chip real estate needs to be distributed across the multiple cores as well as multiple levels of caches. The same constraint applies to power consumption. In the dark silicon, when most of the cores are powered-off, the communication network is expected to be available to maintain on-chip connectivity between the cores and caches. To mitigate the leakage power and area issue, computer architects have moved towards emerging Non-Volatile Memory (NVM) and look at them as alternate memory technologies in the buffers for the NoC routers.

The gain obtained by using NVMs is low leakage power consumption, high density, multi-bit storage capability and excellent scalability. However, by employing NVMs in the router buffers, they suffer from costly write operations and weak write endurance; thereby negatively impacting the performance, energy consumption and lifetime of the buffers and the router as a whole.

Several attempts have been made to counter the power and performance impact in NoC for dark silicon scenario by router power gating, fine-grained power gating of various components in the router, establishing bypass links between the router, by altering frequency of router and/or links based on the utilization of traffic load, by dividing a single plane network into multi-plane physical networks and replacing conventional SRAM based buffers by STT-RAM buffers, etc. In NVM based

designs, limited flit retransmission [83], flit migration [44] and hybrid buffer designs with hierarchical and banked buffers [1] are proposed. These policies mainly reduce the number of writes in the NVM buffer either by (i) using error detection and correction in flit transmission, which limits the number of retransmitted flits; or (ii) using a hybrid buffer along with traffic-based load distribution across SRAM and NVM buffers.

None of the existing policies discusses the lifetime and weak endurance issue of NVM buffers, which is the objective of this thesis. We aim to perform careful management of writes in the NVM technology-based NoC buffers to make them viable replacements of power-hungry SRAM buffers.

Chapter 3

Wear levelling by Static Buffer Assignment to VCs

In this chapter, we discuss the first contribution, which improves the lifetime of non-volatile buffers used in the NoC routers. As the chapter title indicates, we assign a set of buffers to each virtual channel, and these buffers remain permanently assigned to the particular VC throughout the lifetime of the design. Write counts are maintained with each VC, and new packets are assigned to VCs having least write count. This helps in evenly distribution writes across the VCs, thus achieving wear levelling and lifetime improvement. Policies are proposed for both Iso-Capacity and Iso-Area setups. In Iso-Area setup, we take 3 times the number of SRAM buffers/VCs and partition them into different groups. During an interval, a particular group of VCs is selected for allocation. This choice depends on the writes incurred across them. We also explore the hybrid version of the proposals, where some VCs are made of SRAM and some of STT-RAM. In that, the SRAM VCs are used to mitigate the latency degradation of pure STT-RAM counterparts.

3.1 Introduction

As discussed in Chapter 2, in the era of dark silicon, although the cores/caches may be turned off, the communication network, NoC, is expected to be available. The communication network consumes a significant amount of total chip power. Many approaches aim to reduce power consumption by saving power in the buffers of the routers. The buffers in the router are made up of SRAM, which is known to have larger leakage, whereas NVM technologies promise very minimal leakage. This work attempts to reduce the router power consumption by replacing SRAM buffers with NVM buffers.

If we design NoC buffers using STT-RAM technology, then we need to limit the write variation across them to control the endurance issue and enhance the lifetime. Write variation in NoC buffers can be controlled by the virtual channel (VC) allocation policies for NoC routers. If we can control the way in which the buffers are allocated to VC/virtual network (VNet), then the write variation can be controlled and hence lifetime can be improved. Towards this, we propose policies to allocate VC depending on the writes incurred by them during execution.RAM buffers with STT-RAM buffers, in this chapter, we propose an Iso-Capacity as well as Iso-Area VC allocation policies to control write variation of STT-RAM-based NoC buffers.

The major contributions of this work are as follows:

- We present a VC allocation policy, WVAR, to minimize the write variation across the buffers in the VCs in the Iso-Capacity setup.
- In order to mitigate packet latency degradation due to use of long latency NVM buffers, we propose a hybrid policy, Hy-WVAR, that uses an alternate SRAM VC to be used in case of heavy traffic.
- We also propose two new VC-allocation policies: SET-RR and SET-VNet-WC for the Iso-Area scenario.

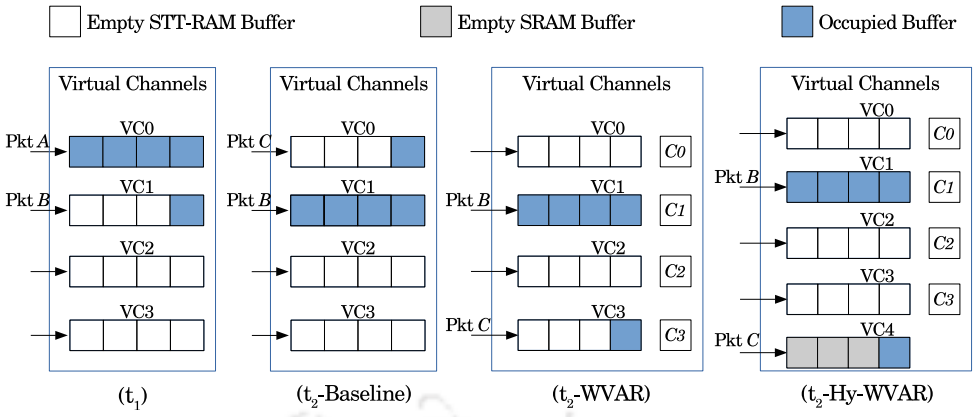


FIGURE 3.1: Virtual network showing VC allocation at different time stamp in baseline and proposed architectures: WVAR and Hy-WVAR.

The rest of the chapter is organized as follows: Background and Motivation are reported in section 3.2. Proposed Iso-Area techniques are discussed in section 3.5. Section 3.9 illustrates the experimental methodology and results and analysis. The overhead analysis due to proposed techniques is reported in section 3.10. Finally, we conclude this chapter in section 3.11.

3.2 Background and Motivation

This section briefly illustrates the basic VC allocation policy in NoC routers.

3.2.1 Basic VC Allocation and Motivation

Figure 3.1 shows an example scenario. At time-stamp t_1, first packet, pkt_A, gets allocated to virtual channel vc_0. When second packet, pkt_B, arrives it gets allocated to vc_1. By the time third packet arrives, i.e. at time-stamp t_2, vc_0 becomes free and hence VC allocator allocates vc_0 to the packet pkt_C (as shown in part *(t_2-Baseline)* of the Figure); whereas channels vc_2 and vc_3 are idle. In this approach, VC allocator chooses the first free VC to allocate packets, thus increasing the number of writes on a particular virtual channel (vc_0 in this case).

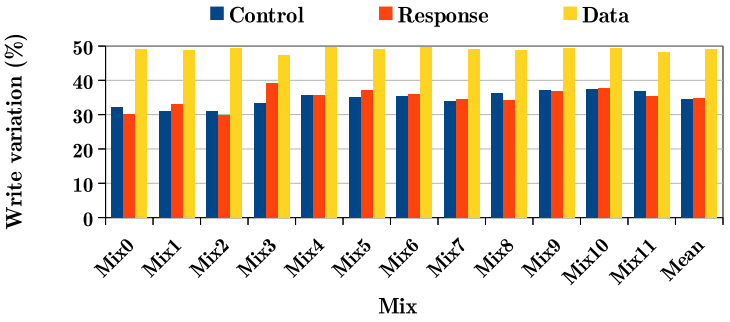


FIGURE 3.2: Write variation in STT-RAM baseline (IC-Base) with 16 core, 4×4 mesh network for different virtual networks. (lower is better)

As described in the illustration, the uneven distribution of writes within the VNet generates the write variation (defined in the next subsection), as shown in Figure 3.2 (Details about the experimental setup and the benchmarks definitions are reported in section 3.9), governed by the baseline VC allocation policy. This allocation policy is fine as long as we have long endurance SRAM buffers. However, with NVM buffers, one needs to consider the buffer's lifetime. Note that the buffer's lifetime is the inverse of the maximum number of writes on that buffer. In particular, the heavily written buffers wear out faster compared to other buffers, leading to a reduction in the overall lifetime. This, in turn, affects the lifetime of the VC and the router.

The Intra-VNet write variation and Lifetime of a buffer is defined in Section 2.3.2.

3.3 Iso-Capacity VC allocation policy: WVAR

In our first proposal, we replace the SRAM buffer with an equal number of STT-RAM buffers. We call this proposal as the Iso-Capacity setup. To uniformly distribute the writes, we maintain counters w_{va} with each input port, virtual network v and virtual channel a. This counter is incremented each time a write happens to a buffer in VC a. If we have these write counts for all VCs within a VNet, we can uniformly distribute the writes. In particular, during the VC allocation stage, among the free VCs, the VC with minimum write counts is allocated to the packet. The method is called as Write VARiation aware VC allocation (WVAR).

Algorithm 1 WVAR Algorithm

1: I : Number of Input ports
2: V : Number of Virtual networks at input port
3: A_v : Set of Virtual channels at virtual network v
4: w_{va} : Number of writes in virtual channel a of virtual network v
5: $w_{va} = w_{va} + 1$, when a new flit is buffered in virtual channel a of virtual network v
6: Store w_{va} for each virtual channel a of virtual network v for every downstream router
7: **for** Every VC allocation stage **do**
8: call $f(WVAR\_VCalloc())$
9: **end for**
10: **f(WVAR_VCalloc())**
11: Find free/idle VCs ($A_v^{'}$) in virtual network v, where $A_v^{'} \subset A_v$
12: **for** $\forall a_i | a_i \in A_v^{'}$ **do**
13: Find $a_j | w_{va_j} = min(w_{va_r}), \forall a_r \in A_v^{'}$ virtual network v
14: **if** $\exists\ j$ and $k \mid w_{va_j} = w_{va_k} = min(w_{va_r}), \forall a_r \in A_v^{'}$ and $j \neq k$ **then**
15: Select VC in Round Robin manner among j, k
16: **end if**
17: Allocate VC
18: **end for**
19: Repeat from line 1 till end of execution

Algorithm: The detailed algorithm is given in Algorithm 1. The parameter I represents the number of input port at a router, V is the number of virtual networks at any input port and A_v represents set of the virtual channel at virtual network v [algo 1, line 1-1]. w_{va} is write counter associated with virtual channel a of virtual network v. The counter gets incremented every time a flit gets written into a VC a of VNet v [algo 1, line 1-1]. For every VC allocation stage, a flit is assigned to a free VC with minimum write count. If there is more than one VC available with minimum write count, allocator chooses VC in round-robin fashion [algo 1, line 10-18]. The process continues till the end of execution [algo 1, line 19].

Example: In the illustration, for convenience, we use C_a as the count for VC a. Consider the buffer allocation at time-stamp t_1 in Figure 3.1, when packet-A and Packet-B are allocated to vc_0 and vc_1, respectively. At time-stamp t_2 when packet-C arrives, in our proposed policy, WVAR, we check an available/free VC having minimum write count and allocate the packet to this particular VC. In this example, write count of vc_3 is the minimum among available VCs, $C_3 = min(C_0, C_2, C_3)$, and thus Packet-C is allocated to vc_3 (as shown in part *(t_2-WVAR)* of the Figure).

3.4 Iso-Capacity VC allocation policy: Hy-WVAR

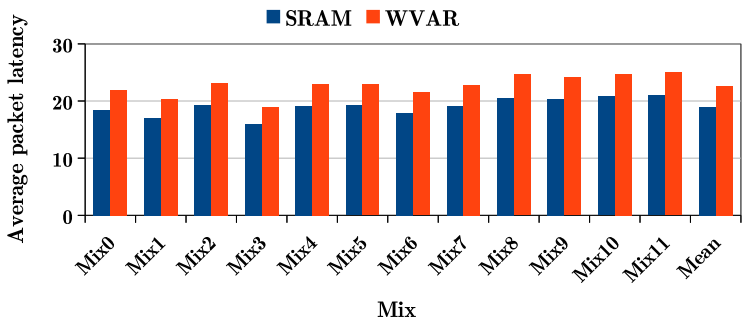


FIGURE 3.3: Average packet latency for different VC-per-VNet values in 4×4 mesh network. (lower is better)

Algorithm 2 Hy-WVAR Algorithm

1: I : Number of Input ports
2: V : Number of Virtual networks at input port
3: A_v : Set of Virtual channels at virtual network v
4: A^{stt} : Set of STT-RAM Virtual channels at virtual network made up of STT-RAM
5: A^{sram} : Set of SRAM Virtual channels at virtual network made up of SRAM
6: Th_v : Threshold network traffic for virtual network
7: Th_{Int} : Number of cycles defining an interval of execution for deciding traffic
8: w_{va} : Number of writes in virtual channel a of virtual network v
9: $w_{va} = w_{va} + 1$, when a new flit is buffered in virtual channel a of virtual network v
10: Store w_{va} for each virtual channel a of virtual network v for every downstream router
11: At the beginning of every interval, Th_{Int}
12: **if** *network traffic* $< Th_v$ **then**
13: Find free/idle VCs ($A_v^{'}$) in virtual network v, where $A_v^{'} \subset A_v^{stt}$
14: **for** $\forall a_i | a_i \in A_v^{'}$ **do**
15: Find $a_j | w_{va_j} = min(w_{va_r}), \forall a_r \in A_v^{'}$ virtual network v
16: *goto line* 33
17: **end for**
18: **else**
19: **for** $\forall a_i | a_i \subset A_v^{stt}$ **do**
20: Find $a_j | w_{va_j} = \max(w_{va_0}, w_{va_1}, ..., w_{vA^{stt}})$ in virtual network v
21: **end for**
22: **if** $\exists$ j and k | $w_{va_j} = w_{va_k} = \max(w_{va_0}, w_{va_1}, ..., w_{vA^{stt}})$ and $j \neq k$ **then**
23: Select VC in Round Robin manner among j, k
24: **end if**
25: Find $A_v^{'} | A_v^{'} \subset A_v^{stt} \cup A_v^{sram} \setminus v_j$
26: **end if**
27: **if** $\exists$ i | $v_i \in A_v^{sram}$ and v_i is $free$ **then**
28: *goto line* 36
29: **else**
30: Find $a_j | w_{va_j} = min(w_{va_r}), \forall a_r \in A_v^{stt}$ and $a_j \in A_v^{stt}$ virtual network v
31: *goto line* 33
32: **end if**
33: **if** $\exists j$ and $k | A_j$ and$A_k \in A_v^{'}$ and $j \neq k$ **then**
34: Select VC in Round Robin manner among j, k
35: **end if**
36: Allocate VC
37: Repeat from line 12 till the end of execution

The earlier technique WVAR performs VC allocation while maintaining uniform writes across all the VCs within a VNet. On account of the high write latency incurred by STT-RAM-based buffers, these routers tend to increase the average

packet latency compared to a SRAM based NoC. It has been experimentally observed that the use of pure STT-RAM buffers increases the latency by 19% (cf. Figure 3.3) over baseline SRAM design in a 4×4 mesh network. In an attempt to rectify this, we propose to use an alternative SRAM VC (as SRAM has lesser write latency) during high network load. In other words, when the network traffic increases beyond a given threshold[1], our proposed method allocates an SRAM VC in place of one heavily written free STT-RAM VC. Traffic analysis is done over equally divided intervals of execution, and the decision to use SRAM is taken at the beginning of every interval. This method of using a combination of SRAM and STT-RAM VCs is called the Hybrid Write VARiation aware VC allocation (Hy-WVAR).

Algorithm: The procedure is detailed in Algorithm 2. The parameters I, V and A_v here are same as Algorithm 1 [algo 2, line 1-3]. Here, A_{sram} are SRAM based virtual channels and A_{stt} are STT-RAM based virtual channels [algo 2, line 4, 5]. The parameter Th_v is used as the predefined threshold [algo 2, line 6] for virtual network v and Th_{Int} is used as a predefined interval for deciding the traffic rate of the network [algo 2, line 7]. The number of writes associated with virtual channel a of VNet v is denoted by w_{va} [algo 2, line 8]. The write count, w_{va}, is increased by one when a flit is written [algo 2, line 9]. In the first interval of application execution, traffic is considered as LOW, and packets are written only into STT-RAM-based VCs. Network traffic is calculated from the second interval onwards. At the beginning of every interval traffic rate will be calculated [algo 2, line 11]. If the rate is LOW, only STT-RAM-based VCs participate in VC allocation stage [algo 2, line 12-17]. Whereas in HIGH traffic scenario, first, SRAM based VC is checked. If it is available, the flit gets allocated to it otherwise free STT-RAM VC with minimum write count gets allocated to the current flit [algo 2, line 18-32]. In order to save leakage power, the SRAM buffers are power gated and not used for VC allocation during LOW traffic intervals. We have considered power gating and wake-up time of SRAM buffers in our simulations. The process continues until the end of execution reaches.

[1]Value of the threshold was computed based on empirical analysis of benchmarks. The detailed analysis with different values of the threshold is given in section 4.2.3.

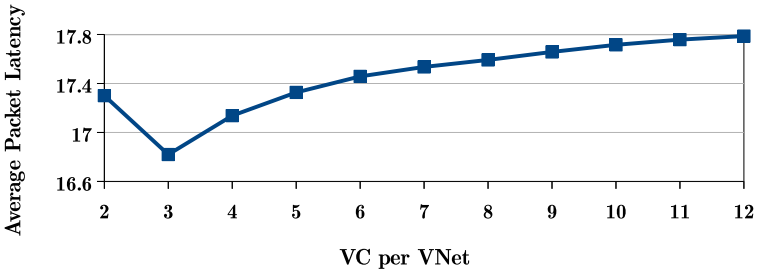


FIGURE 3.4: Average packet latency for different VC-per-VNet values in 4×4 mesh network. (lower is better)

Example: In the illustration (cf. Figure 3.1) consider the buffer allocation at time-stamp t_1 when packets A and B are allocated to vc_0 and vc_1 respectively. At time-stamp t_2, when packet-C arrives, in the proposed policy Hy-WVAR, if the network load is beyond the threshold, we allocate the new packet to the SRAM VC, if it is free and leave the STT-RAM, vc_2, having maximum write counts as idle.

3.5 Iso-Area VC allocation policies

For NoC buffers, any VC allocation policy is fine as long as the buffers are made up of SRAM. However, when we replace them with STT-RAM buffers, we need special care to distribute the writes in order to enhance the lifetime. The techniques presented earlier used in an Iso-Capacity setup, where we replaced SRAM buffers with the same number of STT-RAM buffers. Another approach of replacing SRAM buffers by STT-RAM buffers is in the scenario of Iso-Area.

STT-RAM is at least 4-times denser than SRAM [167]. In our proposals, we use some meta-data counters with each VC, which will occupy space. To take this overhead into account, our iso-area policy uses (conservatively) 3-times the capacity of SRAM while replacing it with STT-RAM buffers. Therefore, we will have three times the number of physical resources to manage. Intuitively the three times resources can be utilized by having three times the number of VCs per VNet.

Every VNet has multiple VCs. If there are very few VCs, the performance gets affected as the packets do not get free VC to allocate. It is, therefore, advisable to have more than one VC per Vnet. It is noticed that the packet latency reduces as we increase VCs per VNet up to a certain number, beyond which the latency again increases. This is because, in each switch allocation stage, only one VC participates from an input port. As a result, the waiting time for a packet increases and hence the packet latency. Figure 3.4 shows that as we increase the VCs per VNet, the latency initially improves, but later it degrades. This motivates us to design newer methods to manage the 3-times resources made available by the Iso-Area setup. In particular, we partition the resources into three sets and propose methods to utilize these sets.

In the first method, we allocate the 3 sets in a round-robin manner and in the second method, we allocate them depending on the write counts in every set.

3.5.1 Iso-Area Baseline (IA-Base)

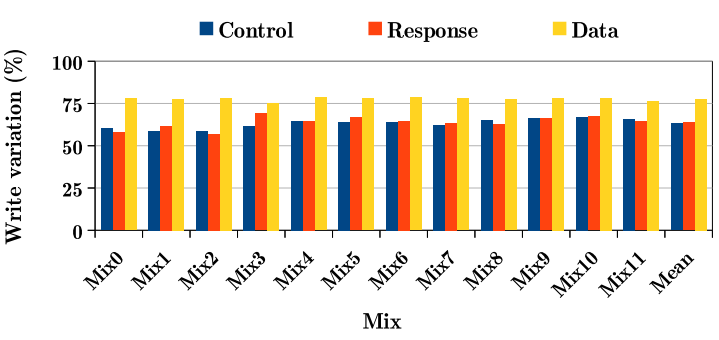


FIGURE 3.5: Write variation in IA-Base with 16 core, 4×4 mesh network for different virtual networks. (lower is better)

The IA-Base uses a similar architecture, as shown in Figure 2.2 with the exception of three times virtual channels in a virtual network as compared to IC-Base. In IA-Base policy, all virtual channels are allotted 3-times per virtual network. Similar to IC-Base, here, the VC Allocator allocates first available VC to an incoming packet even though there is an idle VC with a fewer number of writes (section 3.2.1).

The Figure 3.5 shows write variation in IA-Base in Iso-Area setup. As can be seen, the disparity in the number of writes among virtual channels of a virtual network is more in Data virtual network. Using IA-Base policy leads to more uneven writes in VCs of a VNet than IC-Base (cf. fig 3.2). Hence, this allocation policy increases the number of writes on some VCs, which results in a lesser lifetime of an individual buffer.

In order to rectify unused VCs of a VNet in IA-Base policy, we divide virtual channels into three equal-size sets (buffer set) at each input port. Here, each set will have Control ($VNet_0$), Response ($VNet_1$) and Data ($VNet_2$) virtual networks (Figure 3.6).

3.6 Iso-Area VC allocation policy: SET-RR

While performing an Iso-Area analysis, we take three times the number of buffers when we replace SRAM with STT-RAM. Thus, we have 3-times the number of resources which can be utilized by VCs within each VNet. As seen earlier in Figure 3.5, increasing VCs per VNet has diminishing returns in terms of lifetime, and hence we propose to use the extra resources turn by turn. In particular, we create three separate sets of VNets, each having the same resources as their SRAM counterparts.

Figure 3.6 shows the buffers divided into 3 sets. Each set has 3-VNets, and each VNet is divided into n VCs. The organization of each set is the same as that of the SRAM buffers in the baseline. We divide the execution time into intervals and use one of the three sets in a round-robin fashion. In other words, during one interval, a particular set of VNets is used, and during the next interval, another set of VNets is used. While a VNet is being used for transmission, its VCs are allocated using baseline VC allocation policy (section 3.2.1). At the end of each interval, the partially transmitted packets in the current set of VNets are allowed to be transmitted using their original allocation; whereas any new incoming packets are allocated to the new set of VNets. This overlapped execution lasts only for a few

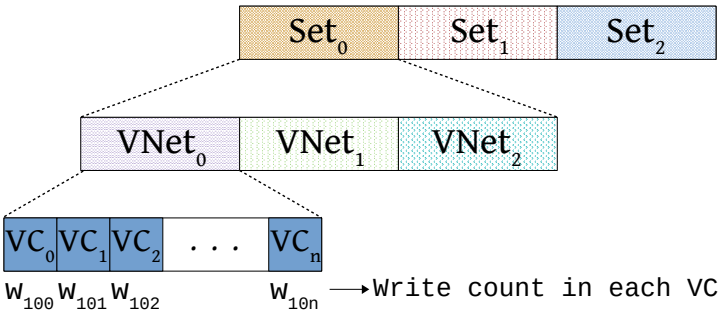


FIGURE 3.6: Distribution of *iso-area* resources across 3-sets. Each set is divided into VNets/VCs.

cycles and consumes only dynamic energy. As STT-RAM has very less leakage, we do not turn-off the sets of VNets that are not in use.

Algorithm 3 SET-RR Algorithm

1: Int_{set} : Number of predefined cycles used for Set selection, defining an interval of execution
2: I : Number of Input ports
3: S : Sets at input port
4: V_s : Number of Virtual networks at input port
5: A_{sv} : Set of Virtual channels at virtual network v of set s
6: $i = 0$
7: At the beginning of every interval, Int_{set}
8: **begin**
9: Select set of VNets s_i — $s_i \in S$
10: Use s_i for packet transmission
11: **for** For every VC allocation stage **do**
12: Use baseline VC allocation policy as given in Section 3.2.1
13: **end for**
14: $i = (i + 1)\%|S|$
15: **end**
16: Repeat from line 7 till the end of execution

Algorithm: The detailed procedure of SET-RR is explained in Algorithm 3. The parameter Int_{set} is a predefined interval used for set selection in SET-RR [algo 3, line 1]. The total number of input ports are defined by I. Here, S represents total number of buffer sets at an input port, V_s number of virtual channel at set s and A_{sv} number of virtual channel at virtual network v of set s [algo 3, line 2- 5]. When application execution begins, set s_0 becomes active and all the incoming packets to the input port get written into it [algo 3, line 6 - 8]. Once the buffer set is selected, the algorithm uses the baseline VC allocation policy at each virtual network to maintain an even distribution of writes across virtual channels [algo 3, line 10- 12]. After Int_{set} cycles, the algorithm chooses s_1 to be active and all incoming packets get written into it. Note that here, algorithm chooses buffer set in round robin

manner [algo 3, line 13]. The process continues till the end of execution [algo 3, line 15].

Example: Figure 3.6 depicts buffer set hierarchy of SET-RR methodology. In the figure, there are three buffer sets (Set_0, Set_1 and Set_2), and each set has n VCs logically grouped into three VNets. Here, buffer selection happens at the topmost level, i.e. set level, in a round-robin manner. If Set_0 is selected in the current interval, then in the next interval, Set_1 will be selected, and the process continues until the end of execution reaches.

3.7 Iso-Area VC allocation policy: SET-VNet-WC

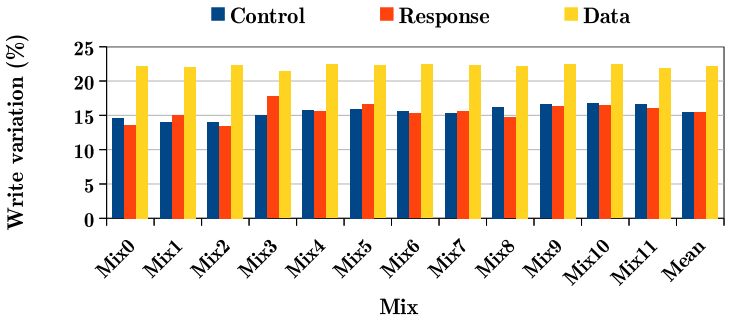


FIGURE 3.7: Write variation in SET-RR with 16 core, 4×4 mesh network for different virtual networks. (lower is better)

When a set is selected for packet transmission, all the VCs of the set participate in it irrespective of the number of writes of an entire set which may increase write variation of VCs in a virtual network across sets. This results in uneven writes in a virtual network across sets. In order to rectify this issue, we propose another method of set selection, SET-VNet-WC.

Figure 3.7 shows the write variation of VNets across the sets, which is around 15.5% in control and response and, 22% in the data virtual network. Note that this is much less compared to variations in baseline IA-Base from Figure 3.5. To rectify this variation and in order to further enhance the lifetime, we propose a

Algorithm 4 SET-VNet-WC Algorithm

1: Int_{set} : Number of predefined cycles used for Set selection, defining an interval of execution
2: I : Set of Input ports
3: S : Set of buffer sets at input port. Let $s_i \in S$
4: V_s : Set of Virtual networks at input port
5: A_{sv} : Set of Virtual channels at virtual network v of set s
6: w_{sv} : Number of writes in virtual network v of set s
7: w_{sva} : Number of writes in virtual channel a of virtual network v of set s
8: $w_{sva} = w_{sva} + 1$, when a new flit is buffered in VC a of virtual network v of set s
9: Store w_{sva} For each VC a of virtual network v of set s for every downstream router
10: Select set s_0 and run application for first interval Int_{set}
11: **for** At the end of every interval, Int_{set} **do**
12: **for** $\forall s \in S \wedge \forall v \in V_s$ **do**
13: $w_{sv} = w_{sv} + w_{sva}$, $\forall a \in A$ //Compute sum of write counts for each VNet in every Set
14: **end for**
//Find VNet, v, with minimum write count among all Sets
15: **for** $\forall v \in V_s$ **do**
16: **for** $\forall s \in S$ **do**
17: $\exists\, i \mid s_i \in S \wedge w_{s_i v} = min(w_{s_j v})$, $\forall s_j \in S$
18: **end for**
19: $v = v_{s_i}$ //VNet v of set s_i will be active in next interval Int_{set}
20: **end for**
21: **for** For every VC allocation stage **do**
22: Call **f(WVAR_VCalloc())** //From algo-1
23: **end for**
24: **end for**
25: Repeat from line 4 till the end of execution

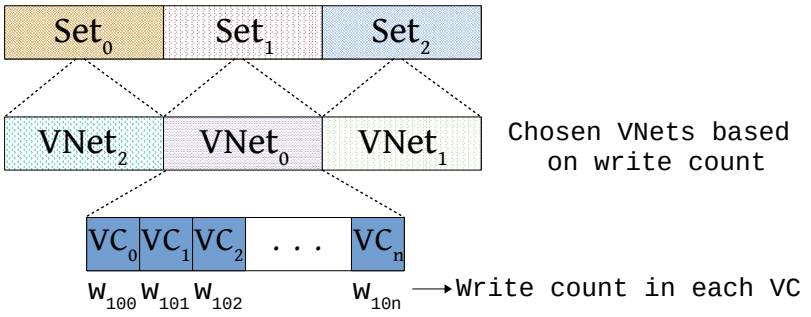


FIGURE 3.8: Selection of VNet from all sets based on write count during k^{th} interval.

method which selects the VNets to be allocated depending on the write counts. The earlier method SET-RR selects a completely new set of VNets in each interval, whereas this method selects each VNet by comparing the writes to that VNet across all sets. For instance, we may select the $VNet_0$ from the first set and the $VNet_1$ from another set. This requires us to compare the counts associated with each VNet before allocating them during each interval. Note that once a VNet is selected, WVAR algorithm is applied inside in order to control write variation across VCs within that VNet. The proposed method is called VNet selection from Sets based on Write Counts-SET-VNet-WC.

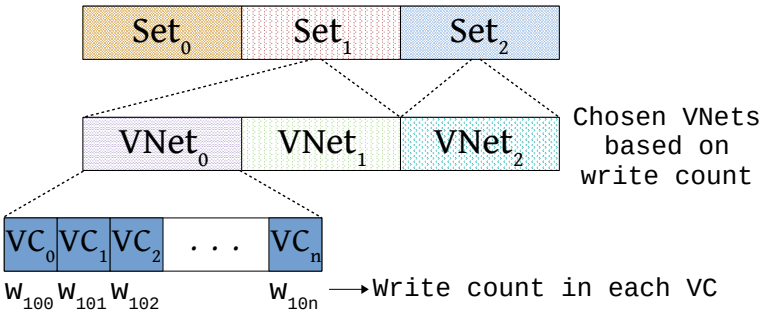


FIGURE 3.9: Selection of VNet from all sets based on write count during $(k+1)^{th}$ interval.

Algorithm: The working approach of SET-VNet-WC is elaborated through Algorithm 4. Similar to SET-RR, the parameter Int_{set} is used as a predefined interval for set selection, I is set of input port of a router, S set of buffer sets at an input port, V_s set of virtual channels at set s and A_{sv} is set of VCs at virtual channel v and set s [line 1 to line 5]. w_{sv} represents number of writes in virtual network v of set s and w_{sva} represents number of writes in virtual channel a of virtual network v of set s [line 6 and 7]. The counter w_{sva} get incremented every time a flit gets written into VC a of VNet v of Set s [line 8]. Algorithm chooses set s_0 to participate in $buffer\ write$ when application execution begins for first interval [line 10]. From second interval onwards, for every interval, write count w_{sv} is calculated for each virtual network in each set [line 11 to 13]. Once the virtual network writes are computed, the Algorithm finds virtual network v_i with minimum write count among all sets in S [line 14 to line 19]. Thus we obtain the VNets across all sets having minimum write counts [line 19]. Unlike SET-RR (reported in 3.6), here, the buffer selection happens at the virtual network level. Once VNets are selected, the algorithm uses Algo 1 to remove unwanted write variation across VCs [line 21 to 23].

Example: Figure 3.8 presents the hierarchy of buffers in SET-VNet-WC methodology. Similar to SET-RR, SET-VNet-WC has three levels of a hierarchy of buffers at each input port. The writes in each VC is denoted by w_{sva} where s is set number, v is virtual network, and a is VC. As shown in the figure, there are three sets at the input port, and each set has n number of VCs logically grouped into three

virtual networks. In Figure 3.8, virtual network $VNet_0$ is selected from Set_1, virtual network $VNet_1$ is selected from Set_2, and virtual network $VNet_2$ is selected from Set_0 for the k^{th} interval Int_{set_k}. With each writes in the Int_{set_k}, the write happens in the buffers associated with the VNets and hence the number of writes increases for active VNets (if any) in the set. The Figure 3.9 shows the active VNets from the sets at $(k+1)^{th}$ interval $Int_{set_{k+1}}$. Here, based on write counts Set_0 does not have any active virtual networks chosen for the interval. Whereas Set_1 have two active virtual networks, $VNet_0$ and $VNet_1$; and Set_2 has one active virtual network: $VNet_2$, for the interval. The selection of active virtual network v is done by comparing the number of writes of the virtual network among all sets. For example, for the selection of $VNet_0$, the values w_{00}, w_{10} and w_{20} will be compared. Since $w_{10} = min(w_{00}, w_{10}, w_{10})$, virtual network $VNet_0$ is active in set Set_1.

3.8 Iso-Area VC allocation policy for Hybrid buffers

As discussed in section 3.7, the proposed method selects a virtual network to be allocated depending on the write counts of each virtual network in each interval. This policy uses pure STT-RAM-based buffers, and hence the packet latency and the performance degrades with respect to SRAM based design. In order to rectify this, we propose another policy SET-VNet-WC-Hy-WVAR. Similar to Hy-WVAR policy, here, each virtual network will have one SRAM based VC. In SET-VNet-WC-Hy-WVAR, once a VNet is selected, Hy-WVAR algorithm is applied inside to control write variation across VCs within that VNet.

3.9 Experimental Evaluation

We evaluate our proposed approaches on a multi-core full system simulator Gem5 [168], with Garnet2.0 [169] as the interconnection network model for NoC performance. We use DSENT [170] for router power analysis at $32nm$ technology. We evaluate

Parameter	Details
Core Count	16, 2GHz
Topology	4 × 4 mesh
L1 I & D cache , L2 cache	64KB, 64KB , 16MB (64B block size)
Router Pipeline	3 stage, 16B flit
Packet Size	control 1 flit, data 5 flits
Virtual Network Count	3 per port with 4 VC per VNet
Virtual Channel Depth	1 buffer for control, 4 buffers for data VC
Counter Size	2B
Buffer Sets	1 (12 Vcs per set) in Iso-Capacity 3 (12 Vcs per set) in Iso-Area

TABLE 3.1: System and interconnect configuration

Parameter	SRAM	Drowsy-SRAM	STT-RAM
write latency (cycle)	1	1	2
read latency (cycle)	1	1	1
read energy/bit (pJ)	0.026	0.026	0.022
write energy/bit (pJ)	0.0363	0.0363	0.182
leakage power (mW)	0.206	0.052	0.067

TABLE 3.2: SRAM and STT-RAM buffer configuration

16 core system with 4 × 4 mesh network with a buffer depth of data VCs 4. We assume a sorted array of write count is maintained, and the head pointer of the array always points to the VC with minimum write count. The detailed system configuration is given in Table 3.1. We use CACTI-STT [171] and NVSIM [172] to get SRAM and STT-RAM latency, read-write energy and leakage power. Table 3.2 shows all the obtained timing and energy parameters. In order to reduce the number of write cycles for STT-RAM, we have brought down the retention time to 10ns [56] [1]. We evaluate our work with SPEC [173] benchmark suites. From the list of SPEC benchmarks, we made 12 multi-programmed workloads for 16 cores. The list of the mix of multi-programmed workload is given in Table 3.3, where each mix has a combination of applications having different write back per kilo instructions. We warm up each multi-programmed workload for 1 billion instructions and after that run for 100 million instructions.

We show the results for (1) Write Variation (calculated by using equation 2.1 and 2.2) (2) Lifetime (3) Total Energy (4) Latency and (5) EDP (Energy Delay Product). The results are given for baseline policy using basic VC allocation policy (cf. section 3.2.1) and the proposed policies: WVAR and Hy-WVAR in Iso-Capacity and SET-RR and SET-VNet-WC in Iso-Area setup. In addition, for Iso-Capacity setup, we have shown comparison analysis with different metrics for WVAR and Hy-WVAR against Hybrid-Drowsy [1]. Improvements in the lifetime and write

Mix	Workloads
0	4×h264ref, 4×libquantum, 4×leslie3d, 4×mcf
1	4×cactusADM, 4×gamess, 4×gromacs, 4×zeusmp
2	4×gamess, 4×gromacs 4×h264ref, 4×mcf
3	4×dealII, 4×cactusADM, 4×leslie3d, 4×zeusmp
4	4×gcc, 4×libquantum, 4×omnetpp, 4×perlbench
5	4×bzip2, 4×hmmer, 4×milc, 4×soplex
6	4×gcc, 4×h264ref, 4×milc,4×perlbench
7	4×bzip2, 4×libquantum, 4×omnetpp, 4×soplex
8	4×dealII, 4×h264ref, 4×sjeng, 4×tonto
9	4×astar, 4×cactusADM, 4×gobmk, 4×namd
10	4×gobmk, 4×namd, 4×sjeng, 4×tonto
11	4×astar, 4×cactusADM, 4×dealII, 4×h264ref

TABLE 3.3: Multi-programmed workloads

variation in Iso-Capacity setup are shown with respect to the baseline, IC-Base, using STT-RAM buffers having the same number of VCs per VNet as the proposed. Whereas, lifetime and write variation in Iso-Area are shown with respect to the baseline, IA-Base, having three times more VCs per VNet as the SRAM baseline. Note that reduction in total energy, EDP gain, and network latency comparison are shown with respect to the baseline, ($SRAM$), using pure SRAM buffers in Iso-Capacity and Iso-Area setup. The reference value for Th_{Int} (defining an interval of execution for deciding the traffic in Hy-WVAR) and Int_{set} (predefined interval used for set selection in SET-RR and SET-VNet-WC) are taken to 2 million cycles.

3.9.1 Results and Analysis

3.9.1.1 Results and Analysis for Iso-Capacity

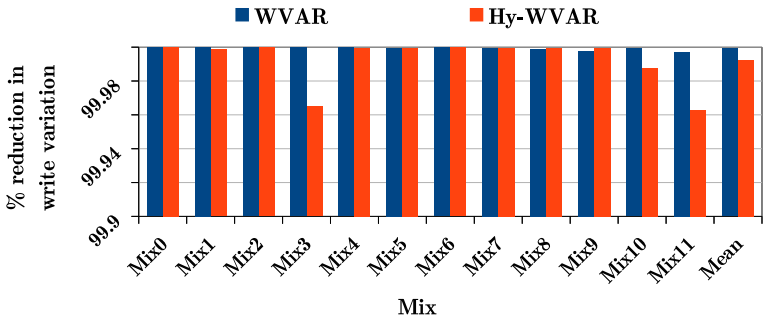


FIGURE 3.10: Percentage reduction in write variation in Control VNet for proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

Write variation: Write variation is calculated and compared individually for each virtual networks for STT-RAM Baseline and STT-RAM Proposed methods. Figure 3.10 shows the reduction in write variation for the control virtual network. It is observed that the average write variation in baseline method is 34% whereas in the proposed methods, it is 0.0001% in WVAR and 0.0003% in Hy-WVAR. The proposed policies thus show a reduction of almost 100% (as seen in the graph).

Write variation for response virtual network is shown in Figure 3.11. Even here, the gap between write variation of baseline and proposed methods is significant. Minimum write variation of STT-RAM Baseline is 30.1% and maximum is 39.2%. For Proposed methods WVAR and HY-WVAR, average write variations are 0.0017% and 0.0033% respectively.

For data virtual network, write variation is maximum for all workloads in STT-RAM Baseline (Figure 3.12). Write variation in STT-RAM Baseline are in the range of 47.3% to 49.6%, and the average is 49%. Overall average write variation in the WVAR, and Hy-WVAR methods are 0.0001% and 0.027% respectively. In the proposed methods, write into VC for each virtual network is more evenly distributed than the default method of baseline, which is evident from almost nil write variation. Uniform distribution of writes across VCs ensures the longer lifetime of the VC buffers.

Lifetime: Similar to write variation, we have done the analysis of buffer lifetime individually for each virtual network since the type of packet differs for each one.

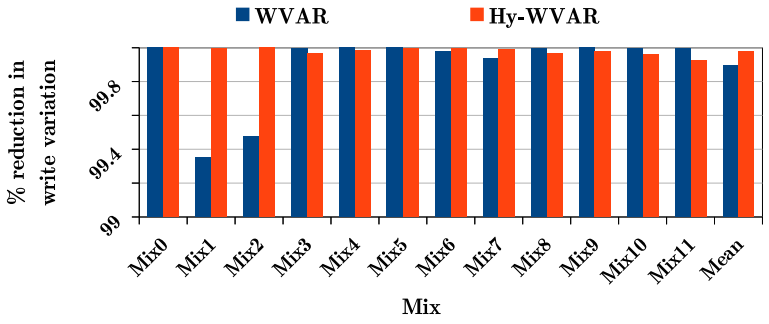


FIGURE 3.11: Percentage reduction in write variation in Response VNet for proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

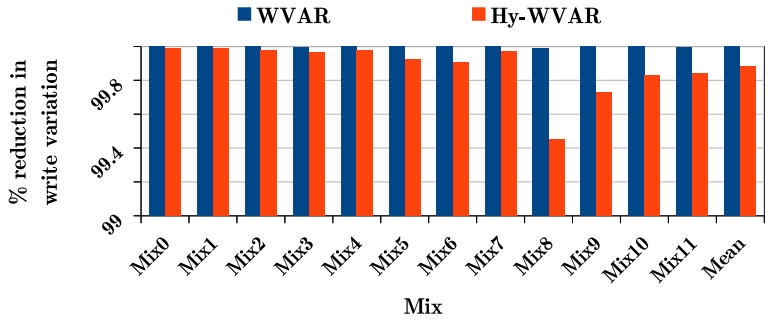


FIGURE 3.12: Percentage reduction in write variation in Data VNet for proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

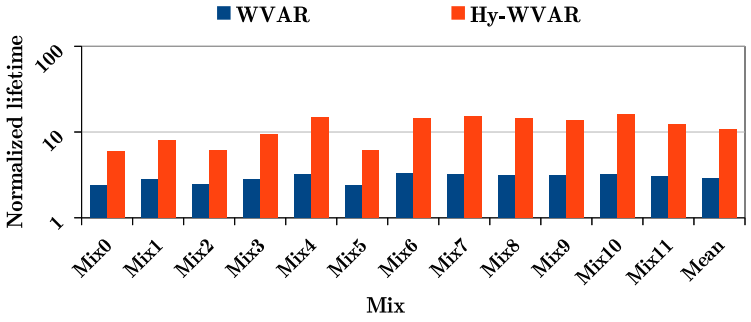


FIGURE 3.13: Normalized lifetime improvement in Control VNet using WVAR and Hy-WVAR policy in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

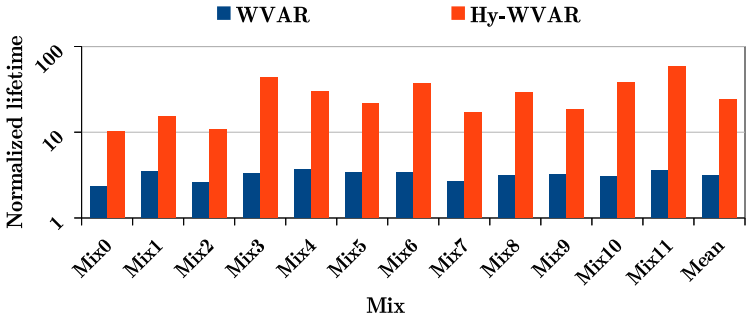


FIGURE 3.14: Normalized lifetime improvement in Response VNet using WVAR and Hy-WVAR policy in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

Figure 3.13 shows lifetime comparison of proposed method for control virtual network. In proposed WVAR technique lifetime is enhanced by 2.4 to 3.3 times. On average, the lifetime is enhanced by 2.9 and 10.6 in WVAR and Hy-WVAR. In response virtual network, Figure 3.14, average life enhancement is 3.1 and 23.9 for WVAR and Hy-WVAR over baseline STT-RAM. For data virtual network,

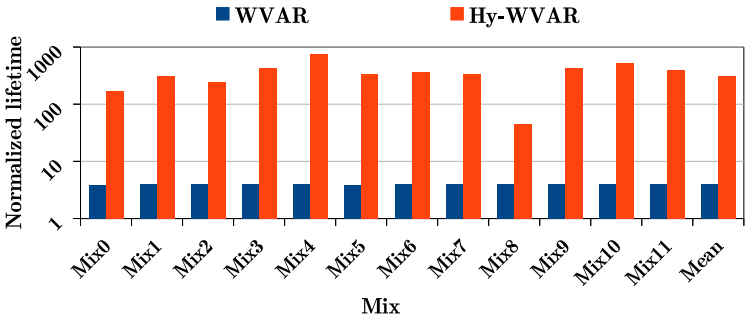


FIGURE 3.15: Normalized lifetime improvement in Data VNet using WVAR and Hy-WVAR policy in Iso-Capacity w.r.t. IC-Base in 4×4 mesh network. (higher is better)

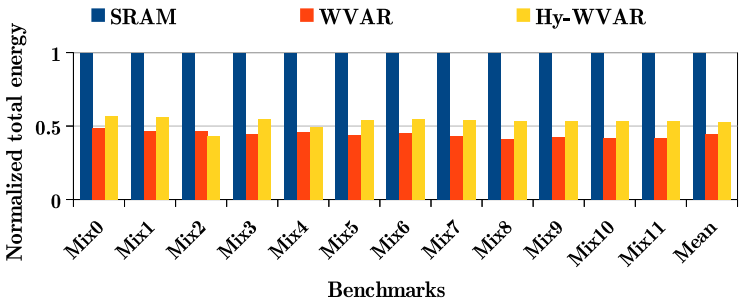


FIGURE 3.16: Normalized total energy of proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

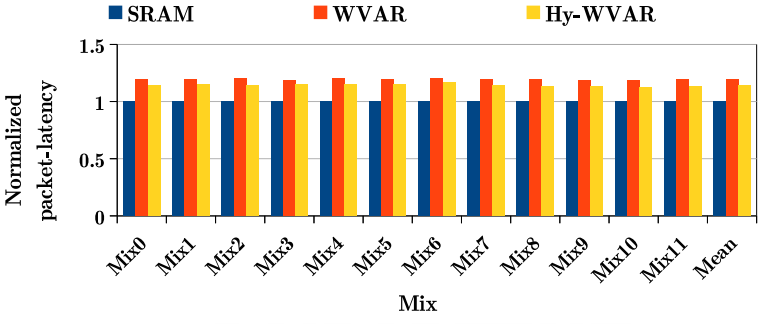


FIGURE 3.17: Normalized packet latency of proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

Figure 3.15, lifetime increases by factor of 3.9 and 308 on average. This huge improvement in a lifetime with Hy-WVAR is due to the use of additional SRAM buffers that reduces the load on STT-RAM buffers. On the other hand, with data VNet, as the data packet has more flits, the impact on lifetime improvement is maximum in the data network.

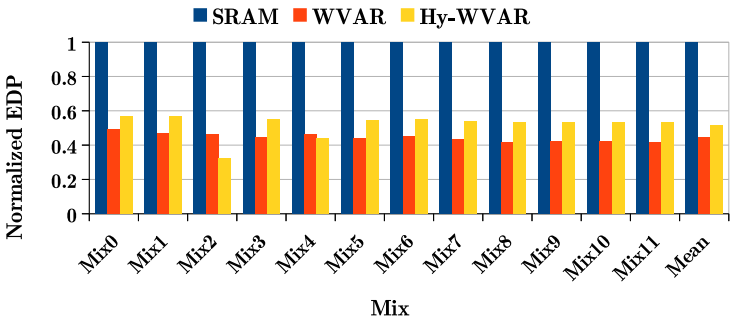


FIGURE 3.18: Normalized EDP gains of proposed policies WVAR and Hy-WVAR in Iso-Capacity w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

Energy analysis: The Figure 3.16 shows the total energy for SRAM baseline and proposed policies. On account of the very less static energy of STT-RAM, we get a significant reduction in total energy in both the proposals. In particular, the total energy is reduced by 55.73% and 47% respectively in WVAR and Hy-WVAR. The writes in STT-RAM require more energy, which increases the dynamic energy; however, lesser static energy gives us overall energy savings.

Packet Latency: The write operation of STT-RAM takes 2 cycles, whereas for SRAM takes only 1 cycle [1]. Note that according to [56], we have used lower retention time in order to reduce write cycles. This impacts packet network latency and hence average packet latency. The Figure 3.17 shows normalized latency in proposed policies with respect to SRAM. WVAR increases packet latency by 19% over SRAM. However, Hy-WVAR attempts to reduce this and achieves the network latency of 14% over SRAM, which is 4.2% improvement over WVAR.

EDP: Slight increase in network latency affects performance. However, the savings in static energy is large and thus leads to an improved EDP. On average we get 55.5% and 48.6% gain in EDP over baseline SRAM [Figure 3.18].

Comparison with existing policy Hybrid-Drowsy [1]:

The contribution in [1] uses a combination of SRAM VCs and STT-RAM VCs, termed as the hierarchical buffer architecture. Here, VCs are power-gated and/or made drowsy based on the traffic load. The proposal uses two SRAM VCs where one VC is always powered-on, and another SRAM VC is kept in low power (drowsy)

mode when not in use. The STT-RAM-based VCs are turned on only during medium to heavy network traffic.

This contribution aims to reduce the power consumption of the NoC routers by using STT-RAM VCs with SRAM VCs. In our proposed methods, we try to reduce write variation and hide the latency by using only one additional SRAM VC (in case of Hy-WVAR), whereas, [1] uses two. The comparison of the Hy-Drowsy policy and our proposed policies with respect to the baseline is shown in Table 3.4 and 3.5.

As seen in Table 3.5, the Hy-Drowsy policy shows an average of 35.57% of write variation across VCs in control virtual networks, 47.22% in response virtual networks and 41.78% in data virtual networks. This impacts the lifetime of STT-RAM-based buffers as a whole. Since the policy uses two SRAM based VCs per virtual network, the average lifetime is 56 times, 23 times and 1109 times in control, response and data virtual network. The lifetime enhancement is because of very few writes in STT-RAM-based VCs since it is only used during high network traffic load along with SRAM VCs. Note that the STT-Writes is calculated per virtual network. It represents the percentage of writes in STT-RAM VCs with respect to overall writes in the VNet. For instance, in response VNet, the percentage of writes in Hy-Drowsy is 0.20% and lifetime is 23 times whereas, the percentage of writes in our proposed Hy-WVAR is 13% which also improves the lifetime of STT based buffers by almost 23 times. This was possible because of 4 STT-RAM VCs and wear levelling of writes across these VCs of proposed approaches. However, Hy-Drowsy avoids writes in STT-RAM VCs by power gating them during normal traffic scenario. Similar results can be seen for control and data virtual networks.

Referring to Table 3.4, in Hy-Drowsy policy, the SRAM VCs balances the latency increase caused by STT-RAM-based VCs. Hence, the latency increase is less with respect to proposed policies. Also, using drowsy SRAM based VCs, the static energy is reduced and hence EDP. The policy shows 28.5% savings in total energy and 28.6% EDP gain with respect to *SRAM* baseline. Our proposed policies use pure STT-RAM VCs hence save more on energy incurring slightly more latency.

Policy	Energy savings (%)	EDP gain (%)	Latency increase (%)
Hy-Drowsy [1]	28.5	28.6	8.09
WVAR	55.74	55.5	19.37
Hy-WVAR	47	48.6	14.4

TABLE 3.4: Energy, EDP and latency comparison with baseline in 4×4 mesh network.

Policy	STT-Writes (%)			Write Variation (%)			Lifetime (times)		
	Ctrl	Res	Data	Ctrl	Res	Data	Ctrl	Res	Data
Hy-Drowsy [1]	0.56	0.20	0.02	35.57	47.22	41.78	56	23	1109
WVAR	100	100	100	0.0001	0.0017	0.0001	2.9	3.13	3.95
Hy-WVAR	28	13	68	0.0003	0.0033	0.027	10.66	23.87	308.2

TABLE 3.5: Write variation and lifetime comparison with baseline in 4×4 mesh network. (Ctrl: Control VNet, Res: Response VNet, Data: Data VNet)

The hybrid-buffer design of [1] has no policy for improving the lifetime of buffers. Thus, our proposed policies not only reduce energy consumption but also improve the lifetime of STT-RAM-based NoC buffers.

3.9.1.2 Results and Analysis for Iso-Area

In the Iso-Area setup, we have three times the number of resources and thus more responsibility to evenly distribute the writes to effectively utilize all the resources.

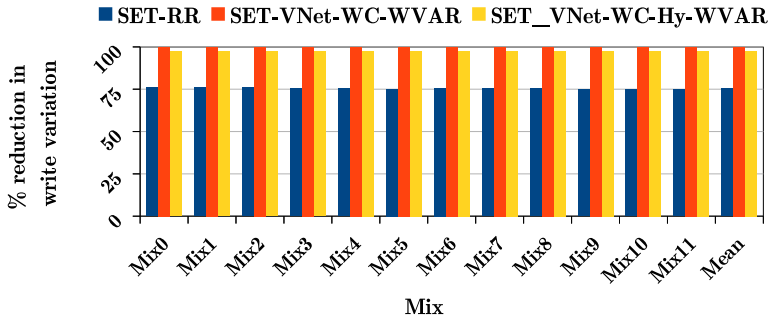


FIGURE 3.19: Percentage reduction of write variation in Control VNet for proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. IA-Base in 4×4 mesh network. (higher is better)

Write Variation: In baseline (IA-Base) policy, there is only one buffer set, and all VCs of a VNet participate in a VC allocation stage. This causes more write variation since the policy use first available VC for packet transmission. Proposed Round robin VNet set selection policy, SET-RR has three sets of VCs and uses

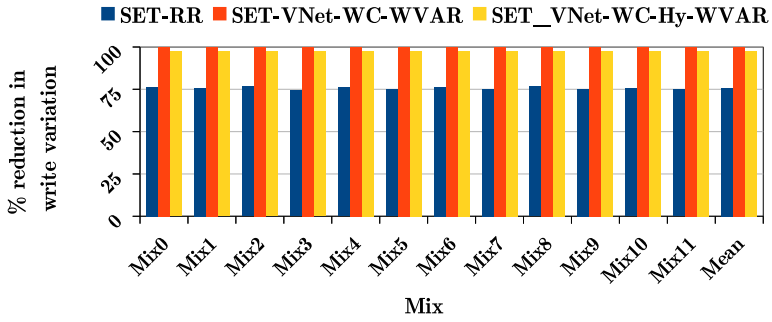


FIGURE 3.20: Percentage reduction of write variation in Response VNet for proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. IA-Base in 4×4 mesh network. (higher is better)

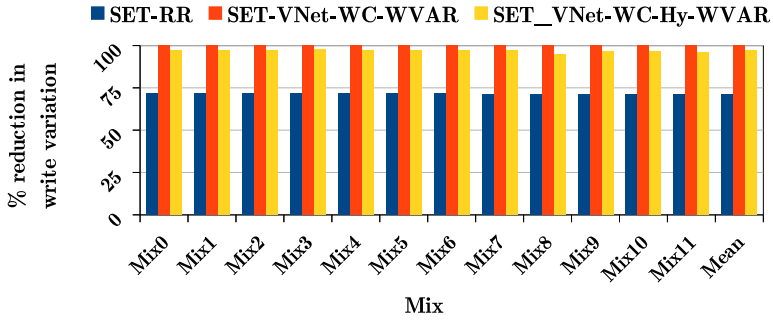


FIGURE 3.21: Percentage reduction of write variation in Data VNet for proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. IA-Base in 4×4 mesh network. (higher is better)

one set for VC allocation in each interval. Usage of buffer sets for VC allocation in round-robin fashion attempts to even out writes of a VNet in three sets. However, this technique does not even out write in VCs of a VNet in a set. The policy SET-VNet-WC further controls the writes to VCs by selecting the appropriate VNet from each set and uses WVAR or Hy-WVAR inside the virtual network. As shown in Figure 3.19, 3.20 and 3.21, compared to the baseline: IA-Base, SET-RR reduces write variation by 55%, 55.3% and 49% in control, response, and data virtual network. The WVAR variant of SET-VNet-WC policy reduces write variation by 99.96% whereas, Hy-WVAR variant reduces it by 97.5% in control virtual network. Similar write variation reduction has been observed in variants of SET-VNet-WC policy for response and data virtual network, which is 99.9% n WVAR variant and 97% in Hy-WVAR variant.

Lifetime: Similar to write variation, we have done analysis on buffer lifetime.

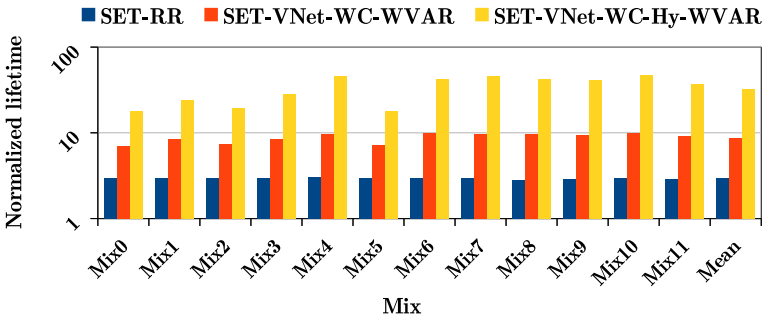


FIGURE 3.22: Normalized lifetime improvement in Control VNet using SET-RR and SET-VNet-WC in Iso-Area policies w.r.t. IA-Base in 4×4 mesh network. (higher is better)

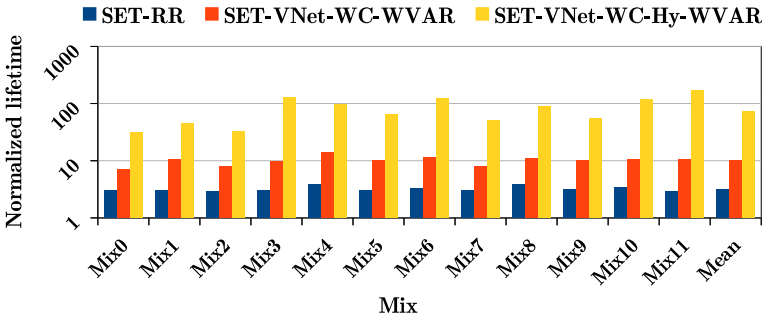


FIGURE 3.23: Normalized lifetime improvement in Response VNet using SET-RR and SET-VNet-WC in Iso-Area policies w.r.t. IA-Base in 4×4 mesh network. (higher is better)

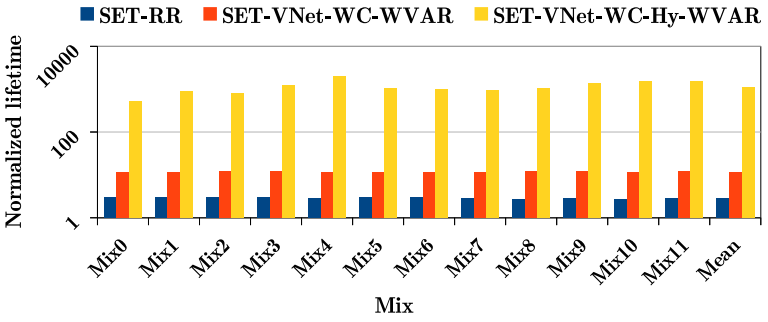


FIGURE 3.24: Normalized lifetime improvement in Data VNet using SET-RR and SET-VNet-WC in Iso-Area policies w.r.t. IA-Base in 4×4 mesh network. (higher is better)

Figure 3.22, 3.23and 3.24 show lifetime comparison of proposed policies in control, response and data VNets for Iso-Area setup. Note that the lifetime graphs are in log scale with base 10. The proposed policy SET-RR enhances lifetime by 2.9, 3.2 and 2.9 times on an average in control, response and data virtual network with respect to IA-Base.

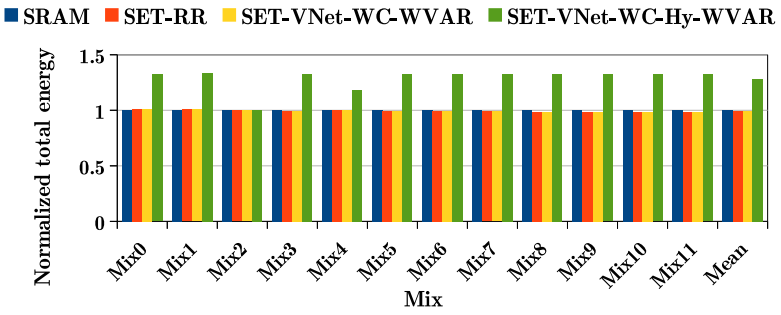


FIGURE 3.25: Normalized total energy of proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

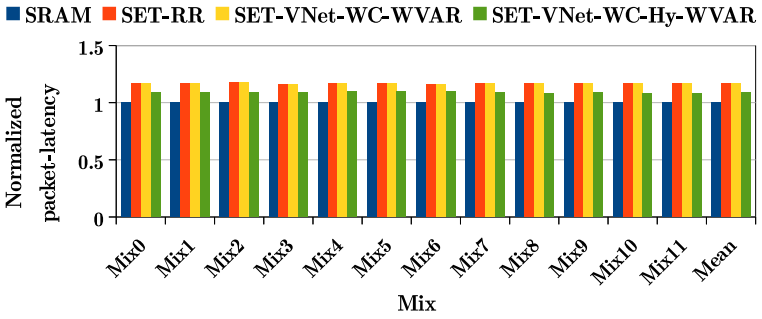


FIGURE 3.26: Normalized packet latency of proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

The SET-VNet-WC policy with WVAR improves lifetime by 8.7 times and with Hy-WVAR improves lifetime by 32 times in control virtual network (Figure 3.22). In response virtual network, Figure 3.23, average lifetime enhancement is 9.9 and 73 times for WVAR and Hy-WVAR variant of SET-VNet-WC over IA-Base. The buffer lifetime increases by a factor of 11.8 and 1093 times in data virtual network (Figure 3.24).

Similar to Iso-Capacity, we get more lifetime improvement in Hy-WVAR variant of SET-VNet-WC policy.

Energy analysis: The Figure 3.25 shows the total energy of proposed policies for Iso-Area setup with respect to SRAM based design. STT-RAM incurs costly writes, which requires more energy and increases the dynamic energy, yet, lesser static energy gives overall energy savings. The total energy is reduced by 0.01% in SET-RR and SET-VNet-WC WVAR variant with respect to SRAM baseline. The

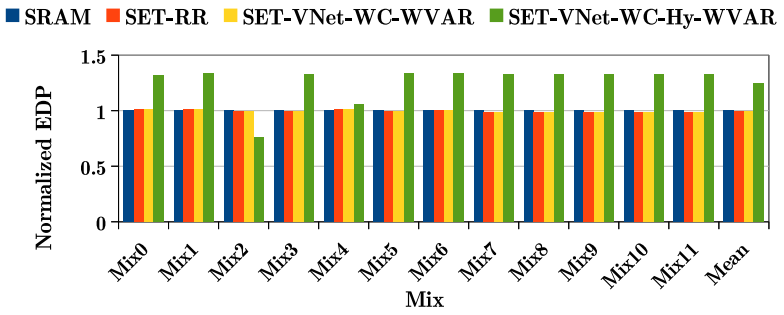


FIGURE 3.27: Normalized EDP gains of proposed policies SET-RR and SET-VNet-WC in Iso-Area w.r.t. SRAM baseline in 4×4 mesh network. (lower is better)

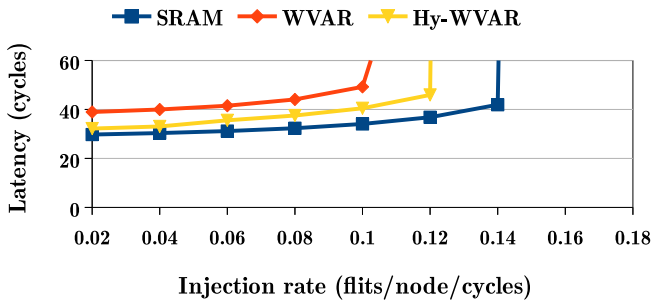
Hy-WVAR variant of SET-VNet-WC policy shows 28% increase in total energy with respect to SRAM baseline. This increase in energy is due to the usage of an additional SRAM VCs.

Packet Latency: The Figure 3.26 shows normalized latency in proposed policies in Iso-Area setup. Similar to WVAR policy, in Iso-Capacity setup, SET-RR and SET-VNet-WC with WVAR give 16.5% increase in packet latency over SRAM. However, the Hy-WVAR variant of SET-VNet-WC policy uses an SRAM VC per VNet and reduces packet latency by 5% with respect to SET-RR policy which is 13% increase over SRAM.

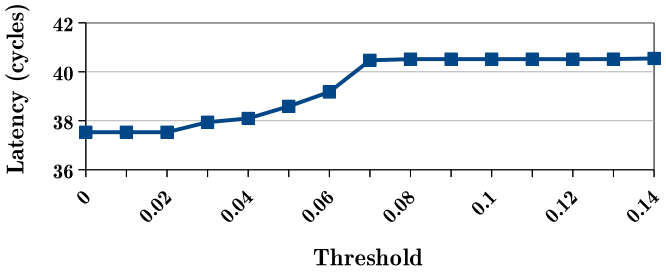
EDP: We get around 0.015% gain in EDP in SET-RR and SET-VNet-WC with WVAR policy with respect to SRAM (Figure 3.27). The Hy-WVAR variant of SET-VNet-WC policy uses an SRAM VC per VNet, which increases leakage power. This results in loss of EDP with respect to SRAM baseline policy. The SET-VNet-WC policy with Hy-WVAR shows 24% EDP loss with respect to SRAM.

3.9.1.3 Analysis using Synthetic Workloads

Latency analysis: To study the impact of proposed policies on the packet latency, we performed simulations on an 8×8 network using different traffic patterns and by varying the injection rate. Figures 3.28(a) and 3.29(a) show performance



(a) Uniform Random

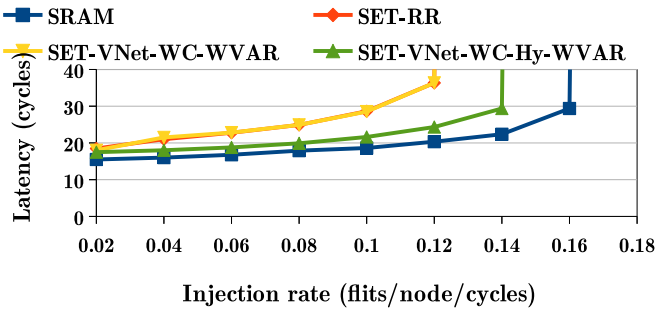


(b) Threshold analysis

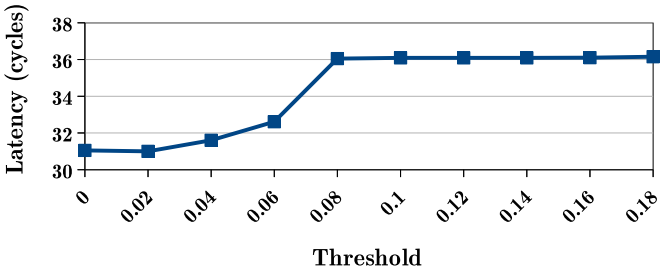
FIGURE 3.28: Performance Comparison and threshold analysis of Iso-Capacity policies with Synthetic Workloads in 8×8 mesh network.

comparison of proposed policies with the baseline under $uniform - random$ synthetic workload. On account of the slow write speed of STT-RAM, the packets take slightly more time to reach the destinations. On Iso-Capacity setup, as expected, the latency of Hy-WVAR policy lies between the latency values for SRAM and WVAR, and both the proposed policies reach saturation earlier than the baseline SRAM. WVAR increases latency by 28% on average w.r.t SRAM, whereas Hy-WVAR shows latency improvement by 14% w.r t. WVAR. On the other hand, with Iso-Area setup, the SET-RR and SET-VNet-WC WVAR variant increases latency by around 28.6% on average, whereas SET-VNet-WC Hy-WVAR variant shows latency to increase by 10.6% over SRAM.

Effect of threshold value: The usage of SRAM based virtual channel in Hy-WVAR policy, both in Iso-Capacity and Iso-Area, depends on the threshold used. Lower threshold value ensures more frequent writes in SRAM based VC. Figures 3.28(b) and 3.29(b) show the effect on network latency using different



(a) Uniform Random



(b) Threshold analysis

FIGURE 3.29: Performance Comparison and threshold analysis of Iso-Area policies with Synthetic Workloads in 8×8 mesh network.

threshold values under uniform-random traffic pattern with fixed injection rate of $0.1 flits/node/cycle$. Higher threshold reduces the number of writes in the SRAM based virtual channel, which results in more average packet latency. Thus, there is a trade-off between packet latency and power. If the threshold is less, latency is less, but it incurs more leakage power due to the frequent use of SRAM. On the other hand, a higher threshold saves leakage but incurs more packet delays.

3.10 Overhead Analysis

Due to the usage of multiple counters, the proposed techniques suffer from the overheads in terms of area, energy, and storage. The counter values used here can be easily truncated when it reaches its maximum limit. This allows us to use small counter of size 2 bytes. This sections will briefly demonstrate all types of

overheads with respect to these counters in Iso-Area and Iso-Capacity setup of different policies.

Counter Storage Overhead: As per the system parameter is given in Table 3.1, a total of 24-byte size of counters are used for each input port. Hence, for a 4x4 network, the total size will be 384 bytes. All these constitute a counter overhead percentage: (1) In the case of Iso-Capacity with respect to IC-Base is 6.25% (WVAR) and 5% (Hy-WVAR) (2) In the case of Iso-Area setup with respect to IA-Base, the counter overhead is 6.25% (SET-VNet-WC) and 5% (SET-VNet-WC-Hy-WVAR), and with respect to IC-Base, the counter overhead is 18.75% (SET-VNet-WC) and 15% (SET-VNet-WC-Hy-WVAR).

Counter Energy Overhead: In our experiment, we have taken SRAM based counters where the leakage is the dominant source of energy consumption. The overhead energy values for Iso-Capacity setup with respect to IC-Base is 6.2%, and for Iso-Area setup with respect to IA-Base is 7.48% of total router power. Note that this energy consumption is marginal compared to SRAM baseline policy as evident from Figure 3.16 and 3.25.

Counter Area Overhead: As the area occupied by the STT buffer is one-fourth in terms of SRAM. In Iso-Area setup, the SET-RR has no counters associated; hence it saves the area by 25% over SRAM baseline design, whereas SET-VNet-WC WVAR variant saves by 6%. However, the Hy-WVAR variant of SET-VNet-WC design saves area by 3% w.r.t SRAM design if one SRAM based VC is used for each virtual network for all sets.

3.11 Summary

NoC is the backbone of communication for today's chip multi-processors and also a significant contributor to the power budget. Specifically, the buffers at the ports consume more power. This chapter proposed to use NVM based STT-RAM buffers

as they promise high density and low leakage. The major drawbacks of STT-RAM are their low write endurance which can affect the lifetime of the buffers.

Towards this the chapter, we proposed Iso-Capacity and Iso-Area based VC allocation policies to reduce write variation across buffers. The Iso-Capacity policies: WVAR and Hy-WVAR reduce write variation across VCs within the VNets by allocating lightly written VCs to newly arriving packets. This reduces the write variation to almost 0%. The lifetime is improved by 3.9 times and 308 times in WVAR and Hy-WVAR respectively.

The Iso-Area proposal divides the three times resources into sets and presents the scheduling of these sets in the VC allocation procedure. A round-robin allocation of sets (SET-RR) improves packet latency but does not help in reducing write variation. This reduces write variation to 55.3% over baseline policy. The other proposal SET-VNet-WC picks up those VNets from across the sets that have least write counts. Write counts are maintained with each VNet, and the selection of VNets to be used for packet transmission is performed at fixed intervals. The policy reduced write variation to almost 0% and improved lifetime by 11.8 times over baseline. The hybrid variant of this policy also reduced write variation and improved lifetime by 1093 times over baseline.

The reduced static energy consumption of NVM technology gives a considerable reduction in total power consumption, leading to EDP gains of 55.5% in Iso-Capacity and maintain the same energy in Iso-Area policies. Thus, careful management of writes in NVM buffers can make them a viable replacement for power-hungry SRAM buffers in on-chip interconnects.

Chapter 4

Wear Levelling by Dynamic Buffer Assignment to VCs

The previous chapter permanently assigned buffers to VCs and subsequently proposed VC allocation depending on write counts. However, even if certain VCs incur less writes, it may be the case that certain buffers inside the VC have undergone several more writes compared to buffers in other lightly written VCs. To nullify this impact, in this chapter, we logically make groups of buffers and assign one or more group of buffers to form VCs. Once all VCs are formed then the allocation of VC to newer packets happens based on write counts. The reconfiguration of VCs using buffer groups is done after regular intervals to guarantee that the buffers inside all VCs have a similar number of writes.

4.1 Introduction

A baseline NoC router has buffers at its input and output ports. The physical links are logically partitioned into virtual networks (VNets) which in turn are divided into virtual channels (VCs). The buffers inside a VC of the VNet store the incoming flits, and depending on the VC allocation policy, the number of writes incurred by the buffers may vary. We quantify this by defining the coefficient of

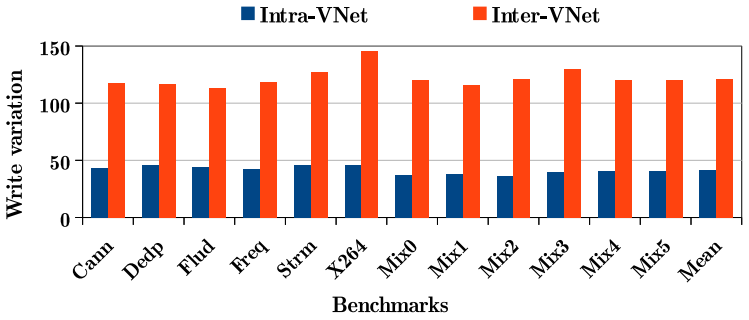


FIGURE 4.1: Write variation in STT-RAM baseline with 16 core, 4×4 mesh network (lower is better).

write variation in Section 2.3.2. Using these definitions and running experiments on a number of benchmark programs, we obtain results, as shown in Fig. 4.1. Details about the experimental setup and the definition of the benchmark are reported in Section 4.5. From the figure 4.1, it can be seen that Intra-VNet variation is quite significant across VCs belonging to the same VNet. The variation can be controlled by evenly distributing the writes across the VCs by using appropriate VC allocation policies. In the previous chapter(3), we presented VC allocation policies that significantly reduce the write variation across VCs within a VNet.

However, there are still scenarios when buffers assigned to some VNets incur lesser write compared to some other VNets generating Inter-VNet write variation, as shown in Fig. 4.1. In other words, if we can manage the write variation across all the buffers at input ports instead of the buffers within each VNet, we may be able to further improve the lifetime of the NVM buffers. Towards achieving this, the major contributions of this chapter are:

- We present VC allocation policies based on write counts of buffers across different VNets. This will reduce write variation across VNets termed as *Inter-VNet* write variation.
- Two policies: Dy-WVAR and Hy-Dy-WVAR are presented to reduce *Inter-VNet* write variation.
- We also presented a comparative analysis with an existing technique Hybrid-Drowsy [1] and the results show the significant improvement.

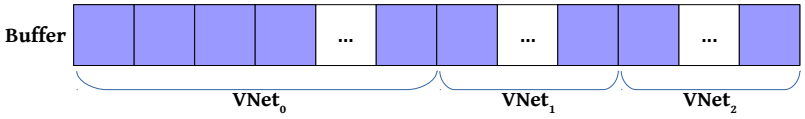


FIGURE 4.2: Static buffer allocation

- We present standalone parameter based analysis for the *Intra-VNet* and *Inter-VNet* policies.

The chapter is organized as follows: Background and Motivation are reported in section 4.2. Proposed Dynamic write variation techniques are discussed in section 4.3 and 4.4. Section 4.5 illustrates the experimental methodology and results and analysis. Storage overhead analysis is reported in section 4.6. Finally, we conclude this chapter in section 4.7.

4.2 Background and Motivation

Previously described policy, WVAR (c.f. section 3.3) and Hy-WVAR (c.f. section 3.4), allocates VCs depending on the variation in the number of writes in each VC. This method helps in reducing the variation inside a VNet over the baseline VC allocation policy. However, there could be write variation across VNets as the number of writes happening in different VNets vary with application profile. Figure 4.1 shows this write variation across VNets, which is still responsible for reducing the lifetime of NVM buffers. WVAR policy guarantees equal write distribution within a VNet. Our next goal is to reduce this variation across different VNets.

Figure 4.2 shows the sequential allocation of buffers to VNets. Once this allocation of buffers to VNets is done, it remains unchanged throughout the lifetime of the system. If we can make this assignment dynamic, i.e. control the assignment of buffers to the VNets taking into account the write variation among them, we will be able to further improve the lifetime of all the buffers.

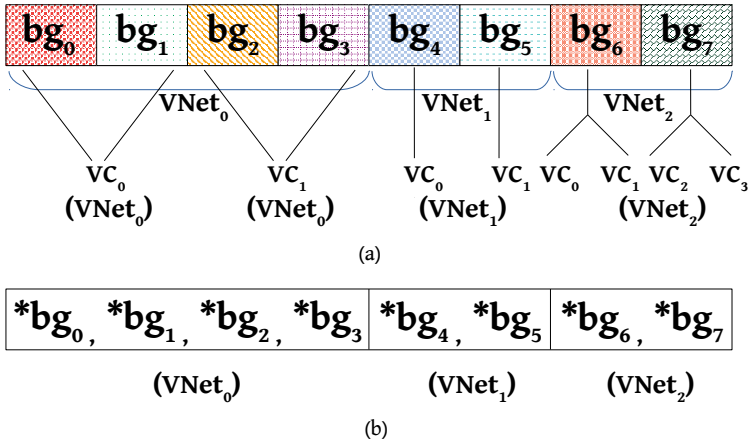


FIGURE 4.3: Buffer allocation for Dy-WVAR policy at timestamp t_1.

4.2.1 Buffer Groups

For controlling the write variation across VNets, we partition the set of buffers into smaller groups called *buffer groups*. The assignment of a buffer to VNets is done at the granularity of the buffer groups instead of sequential fixed allocation. The buffer groups are reshuffled among the VNets at run-time depending on their write counts. This reallocation is done at fixed intervals during execution. Note that every VNet may require one or more buffer groups, which will then be distributed among the VCs belonging to that VNet. This proposed policy is called Dynamic-WVAR (Dy-WVAR), which dynamically assigns write variation aware buffer groups to VNets and additionally performs WVAR within each VNet.

Figure 4.3 shows the buffer groups and their allocation to VNets at a given time instant (t_1). Each buffer group is shown with a different shade and the VNet associated to the buffer group(s) (figure 4.3(a)). The figure also shows the logical assignment of buffer groups to VCs within each VNet. Some VNets have VCs requiring multiple buffer groups, while certain other VNets have VCs sharing a buffer group. For example, a data VNet will use multi-buffer groups as it received packets with multiple flits; whereas control VNets will have VCs that use a single buffer. Formally, the VNets can be seen as an array of pointers pointing to the assigned buffer-group, as shown in figure 4.3(b).

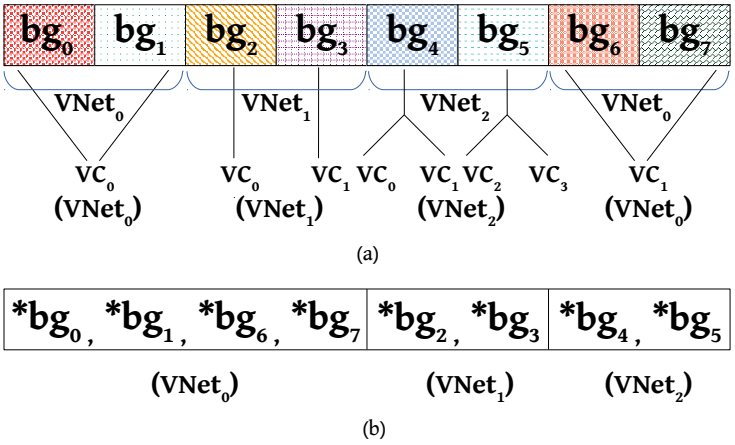


FIGURE 4.4: Buffer allocation for Dy-WVAR policy at timestamp t_2.

4.3 Inter-VNet policy: Dy-WVAR

Write counts are associated with each buffer group as well as with each VNet. With a given assignment, the execution continues over an interval. At the end of an interval, depending on the write counts of the buffer groups and the write counts of the VNets, reallocation of buffer groups to VNets takes place. In particular, buffer groups having more write counts are allocated to VNets incurring less write counts. This helps in balancing the number of writes and reducing write variation. Figure 4.4(a) shows the new assignment of buffer groups to VNets at a later timestamp (t_2). Figure 4.4(b) shows the new pointer assignments.

At the end of each interval, the reallocation must take place. For this, we need the sorted order of the write counts for the buffer groups as well as for the VNets. This sorted order can be obtained in the background towards the end of each interval. The reallocation, however, will require to stall the router pipeline. This is because, during reallocation, we need to update the VNet pointers to point to the new group of buffer assigned to it. Here, we need one extra set of buffer group as swap space; and need $O(n)$ time to reallocate the pointers of VNets to the new set of buffer groups (n is the number of buffer groups at the input port). This additional swap space and the stall overhead are taken into account in our simulations. The detailed process is elaborated in Algorithm 5.

Algorithm 5 Dy-WVAR Algorithm

1: I : Number of Input ports
2: V : Set of Virtual networks at input port
3: A_v : Set of Virtual channels at virtual network v
4: Int : Predefined interval
5: n : Number of Virtual networks at input port
6: G : Number of buffer groups
7: g_i : Number of buffer groups required by virtual network i (Within each VNet these g_i are sequentially allocation to VCs.)
8: wv_i : Number of writes in virtual network i, $0 \leq i < n$
9: wb_j : Number of writes in buffer group j, $0 \leq j < G$
10: In the beginning of execution, sequentially assign the buffer groups to the virtual networks and run the program for Int number of cycles.
11: Increment the counter wv_i on every write to the virtual network i
12: Increment the counter wv_i on every write to the buffer group j
13: At the end of Int cycles **do**
14: Sort array wv in ascending order
15: Sort array wb in descending order
16: Starting from index-0 of both arrays, successively assign buffer groups to VNets. {The sorting guarantees that VNet incurring maximum writes
17: will get buffer group incurring least writes.}
18: **for** Every VC allocation stage **do**
19: call f(WVAR_VCalloc()) { as given in Algorithm 1 }
20: **end for**
21: Repeat from line 13 till end of execution

4.3.1 Algorithm

Apart from the parameters of Algorithm 1, we use some additional ones in the proposed method. G denotes the total number of buffer groups formed by grouping two or more buffers [algo 5, line 6]. From these groups, for every virtual network i assign g_i number of groups depending on its requirement [algo 5, line 7]. Within the VNet, the g_i groups are distributed among the VCs. We use two arrays of counters: wv and wb. Here wv maintains the write counts incurred by every virtual network [algo 5, line 8] and wb maintains the write counts incurred by every buffer group [algo 5, line 9].

Initially assign the buffer groups sequentially to every VNet and run the application for the given interval Int [algo 5, line 10]. The write counters are updated during the interval [algo 5, line 11, 12]. Towards the end of the interval, the write

count arrays wv and wb are sorted in ascending and descending order, respectively [algo 5, line 14, 15]. At the end of the interval, we need to reassign the buffer groups incurring maximum write to the VNet incurring the least number of writes. This is done with the help of the two arrays wv and wb [algo 5, line 16]. Note that while updating the contents of the VNets, we need to stall the router pipeline. Here, the content to be written in the VNet array using the sorted arrays wv and wb is known; therefore, we need $O(n)$ time for the task. We also need an additional constant number (one in our case) of extra buffer group as swap space. The stall required at the end of each interval is taken into account in our simulation environment.

After the buffer group reallocation, run the application for another interval. During the interval, for every VC allocation instance, Dy-WVAR policy uses the WVAR algorithm (cf. function given in Algorithm 1) [algo 5, line 18 - 20].

4.3.2 Working Example

The figure 4.3 shows VC-buffer mapping scenario in time interval Int_1 (time t_1). This is the scenario of very first interval when application execution begins. The buffer group bg_0 and bg_1 is mapped to vc_0 of $VNet_0$, bg_2 and bg_3 to vc_1 of $VNet_0$ and so on. During the current interval, the packets coming for VC vc_i will get written to its associated buffer group, say bg_j, incrementing the number of writes to this buffer group in particular. Later in the interval Int_2 (time t_2), the mapping of VCs with buffer groups changes due to the disparity in the number of writes among buffer groups. The VC, which has the maximum number of writes during the previous interval, gets assigned to the buffer group with minimum write count. Figure 4.4 shows the scenario at a later timestamp t_2. Due to this methodology, the packets are evenly distributed among buffer cells, increasing the lifetime of individual buffers.

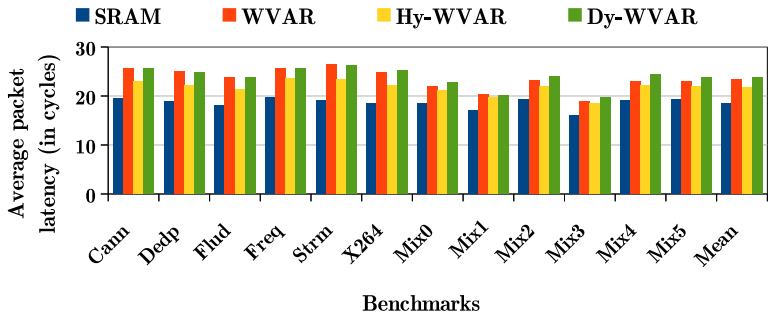


FIGURE 4.5: Average packet latency in SRAM, WVAR, Hy-WVAR and Dy-WVAR policy in 4×4 mesh network (lower is better).

4.4 Inter-VNet policy: Hy-Dy-WVAR

The above section discussed the Dy-WVAR policy which performs dynamic allocation of buffer groups and helps in distributing the write evenly across VNets. However, it still inherits the drawback of WVAR, in that the use of all STT-RAM buffers tends to slow down the network. Towards resolving this, similar to the proposed Hy-WVAR (cf. section 3.4), we propose the use of an additional SRAM channel to be used in case of heavy network traffic. We rightfully call this policy Hybrid Dynamic WVAR, Hy-Dy-WVAR. As STT-RAM-based buffers have limited write endurance, they follow the allocation policy as mentioned in algorithm 2. We have additional buffers allocated to each VNet that are made up of SRAM. The policy assigns buffers to VNets as per Dy-WVAR (cf. Algorithm 5) and within the VNet, depending on the traffic, the additional SRAM VC is used over the interval.

The procedure is similar to Algorithm 5 with the difference that during VC allocation stage [algo 5, line 17], invoke Algorithm 2 instead of the function WVAR_VCalloc() from Algorithm 1.

4.5 Experimental Evaluation

We evaluate our proposed approaches on a full system Gem5 [168], a multi-core simulator, with Garnet2.0 [169] as the interconnection network model for NoC

Parameter	SRAM	Drowsy-SRAM	STT-RAM
write latency (cycle)	1	1	2
read latency (cycle)	1	1	1
read energy/bit (pJ)	0.026	0.026	0.022
write energy/bit (pJ)	0.0363	0.0363	0.182
leakage power (mW)	0.206	0.052	0.067

TABLE 4.1: SRAM and STT-RAM buffer configuration

Parameter	Details
core count	16, 64, 2GHz
topology	4×4 mesh, 8×8 mesh
L1 I & D cache , L2 cache	64KB, 64KB , 16MB (64B block size)
router pipeline	3 stage, 16B flit
Packet size	control 1 flit, data 5 flits
virtual network count	3 per port with 4 VC per VNet
virtual channel depth	1 buffer for control, 4 buffers for data VC
counter size	2B
buffer group count	6 (4 buffers each group)

TABLE 4.2: System and interconnect configuration

performance. We use DSENT [170] for router power analysis at $32nm$ technology. We evaluate 16 core system with 4×4 and 64 core system with 8×8 mesh network with a buffer depth of data VCs 4. The detailed system configuration is given in table 4.2. We use CACTI-STT [171] and NVSIM [172] to get SRAM and STT-RAM latency, read-write energy and leakage power. Table 4.1 shows all the obtained timing and energy parameters. We evaluate our work with PARSEC [174] and SPEC [173] benchmark suites. The acronym used for benchmarks is given in table 4.3. From the list of SPEC benchmarks, we made 12 multi-programmed workloads for 16 cores and 6 for 64 cores. The list of the mix of multi-programmed workload is given in table 4.4. We warm up each multi-programmed workload for 1 billion instructions and run for 100 million instructions.

Parsec Benchmarks
Canneal (Cann), Dedup (Dedp), Fluidanimate (Flud), Freqmine (Freq), Streamcluster (Strm), X264 (X264)
SPEC Benchmarks
astar (as), bzip2 (bz), cactusADM (ct), calculix (cl), dealII (dl), gamess (ga), gcc (gc), gobmk (go), gromacs (gr), h264ref (hr), hmmer (hm), leslie3d (ls), libquantum (lb), mcf (mf), milc (ml), namd (na), omnetpp (om), perlbench (pb), sjeng (sj), soplex (sp), tonto (to), zeusmp (ze)

TABLE 4.3: Benchmark acronym

Mix	16 cores	Mix	64 cores
0	4×hr, 4×lb, 4×ls, 4×mf	**0**	8×hr, 8×mf, 8×lb, 8×ls,
1	4×ct, 4×ga, 4×gr, 4×ze		8×ct, 8×ze, 8×ga, 8×gr
2	4×ga, 4×gr 4×hr, 4×mf	**1**	8×hr, 8×mf, 8×dl, 8×ls,
3	4×dl, 4×ct, 4×ls, 4×ze		8×ct, 8×ze, 8×ga, 8×gr
4	4×gc, 4×lb, 4×om, 4×pb	**2**	8×bz, 8×gc, 8×hm, 8×lb,
5	4×bz, 4×hm, 4×ml, 4×sp		8×ml, 8×om, 8×pb, 8×sp
6	4×gc, 4×hr, 4×ml,4×pb	**3**	8×as, 8×gc, 8×go, 8×hm,
7	4×bz, 4×lb, 4×om, 4×sp		8×lb, 8×ml, 8×om, 8×pb
8	4×dl, 4×hr, 4×sj, 4×to	**4**	8×dl, 8×sj, 8×hr, 8×to,
9	4×as, 4×cl, 4×go, 4×na		8×as, 8×na, 8×go, 8×cl
10	4×go, 4×na, 4×sj, 4×to	**5**	8×as, 8×cl, 8×dl, 8×go,
11	4×as, 4×cl, 4×dl, 4×hm		8×hm, 8×na, 8×sj, 8×to

TABLE 4.4: Multi-programmed workloads

4.5.1 Evaluation Metrics

Improvements in the lifetime and write variation are shown with respect to the baseline, (*STT-Base*), using STT-RAM buffers having the same number of VCs per VNet as the proposed techniques. Reduction in total energy, EDP gain, and network latency comparison are shown with respect to the baseline, (*SRAM*), using pure SRAM buffers. The reference value for Th_{Int} and Int are taken 2 millions cycles.

Results and Analysis is presented in the following subsections.

4.5.2 Write Variation

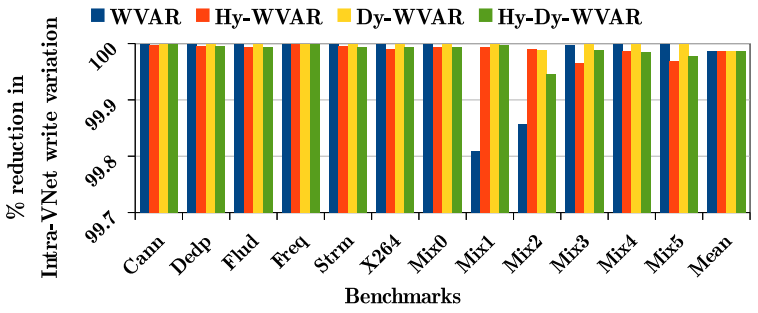


FIGURE 4.6: Percentage reduction in Intra-VNet Write variation for proposed policies w.r.t. STT-Base in 4×4 mesh network (higher is better).

Reduction in Intra-VNet write variation for all proposed methods is shown in Figure 4.6 w.r.t. to STT-Base technique. Table 4.5 shows the reduction in write variation of all policies with respect to each other and baseline for a $4x4$ mesh as

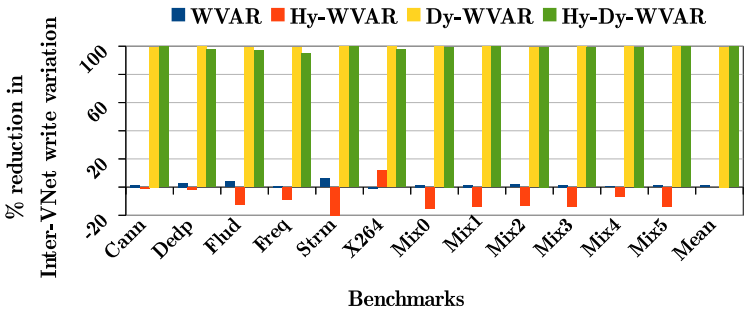


FIGURE 4.7: Percentage reduction in Inter-VNet Write variation for proposed policies w.r.t. STT-Base in 4×4 mesh network (higher is better).

well as $8x8$ mesh setup. Note that negative value in the table (row 2, 3, 7, 8 and 10) implies the increase of write variation. All the four policies: those targeting Intra-VNet (WVAR and Hy-WVAR) and those targeting Inter-VNet (Dy-WVAR and Hy-Dy-WVAR) reduce the Intra-VNet write variation beyond 99% (row 1, 6). The average value of write variation in the baseline is around 43% and, the proposed policies bring this value down to almost 0%. In the proposed methods, write into VC for each virtual network is more evenly distributed than the default method of baseline, which is evident from almost nil write variation. Uniform distribution of writes in VNet across VCs ensures a longer lifetime of the VC buffers.

Reduction in Inter-VNet write variation for all proposed methods is shown in Figure 4.7 w.r.t. STT-Base technique. As expected the Inter-VNet policies Dy-WVAR and Hy-Dy-WVAR reduce the write variation to almost 0% over the baseline (row 2 and 7), WVAR (row 3 and 8) and Hy-WVAR (row 4 and 9). However, Hy-Dy-WVAR shows similar improvement over Dy-WVAR policy (row 5 and 10). The policies targeting only Intra-VNet write variation do not affect the Inter-VNet variation as they are not capable of handling the same. WVAR does not affect the Inter-VNet variation. In some cases, Hy-WVAR is seen to increase the Inter-VNet write variation. This is because of using the additional SRAM VC which changes the allocation pattern of STT-RAM VCs across VNets depending on the traffic load. This increase is however marginal (also seen in table 4.5, row 2, 3, 7 and 8). In all cases, the proposed Inter-VNet policies improve the write variation.

Network size	Reference policy	Write variation	WVAR	Hy-WVAR	Dy-WVAR	Hy-Dy-WVAR	Row
4X4	STT-Base	Intra-VNet	99.97	99.99	99.99	99.99	1
		Inter-VNet	1.15	-10.43	99.31	99.57	2
	WVAR	Inter-VNet	-	-12.20	99.30	99.56	3
	Hy-WVAR	Inter-VNet	-	-	99.35	99.60	4
	Dy-WVAR	Inter-VNet	-	-	-	0.31	5
8X8	STT-Base	Intra-VNet	99.64	98.99	99.65	99.20	6
		Inter-VNet	0.20	-4.96	99.44	97.04	7
	WVAR	Inter-VNet	-	-5.17	99.43	97.04	8
	Hy-WVAR	Inter-VNet	-	-	99.46	97.18	9
	Dy-WVAR	Inter-VNet	-	-	-	-3.88	10

TABLE 4.5: Percent reduction in write variation for all policies.

4.5.3 Relative Lifetime

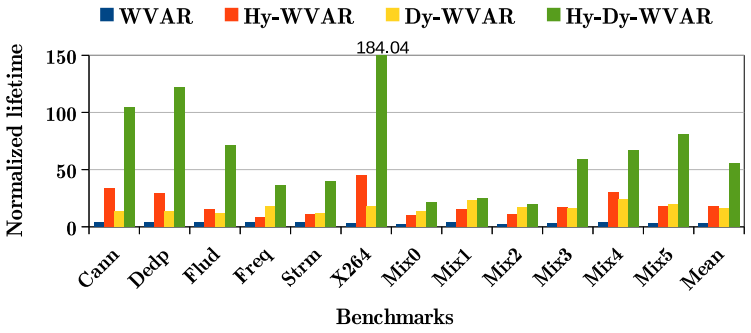


FIGURE 4.8: Normalized buffer lifetime in proposed policies w.r.t. STT-RAM baseline in 4×4 mesh network (higher is better).

Figure 4.8 shows lifetime comparison of proposed methods against STT-Base. Table 4.6 lists the improvement values for all proposed policies with respect to each other and the STT-Base. Note that the table contains the geometric mean of all simulations run on benchmarks mentioned in tables 4.3 and 4.4. Compared to STT-Base, the proposed methods largely improves the lifetime (row 1 and 5). These improvements are basically due to the reduction of Intra-VNet and Inter-VNet write variation by the proposed methods as reported in table 4.5 (table 4.5, row 1, 2, 6 and 7). In particular, over STT-baseline, the lifetime improvements values are 19.92 times and 55.5 times for Dy-WVAR and Hy-Dy-WVAR in 4×4 mesh network and 12.6 times and 55.68 times in 8×8 mesh network. However, we observed that in Hy-WVAR and Hy-Dy-WVAR, the lifetime enhancement is more than the WVAR and Dy-WVAR (row 2, 4, 6 and 8). For instance, lifetime improves by 3.24 times and 19.3 times for WVAR and Hy-WVAR, respectively in 4×4 mesh network.

Network size	Reference policy	WVAR	Hy-WVAR	Dy-WVAR	Hy-Dy-WVAR	Row
4X4	STT-Base	3.24	19.3	19.92	55.5	1
	WVAR	-	5.95	6.2	17.11	2
	Hy-WVAR	-	-	1.03	2.9	3
	Dy-WVAR	-	-	-	2.8	4
8X8	STT-Base	2.8	13.1	12.6	55.68	5
	WVAR	-	4.69	4.49	19.86	6
	Hy-WVAR	-	-	0.96	4.24	7
	Dy-WVAR	-	-	-	4.42	8

TABLE 4.6: Lifetime comparison (in times) for all policies.

In 8×8 mesh network, the lifetime enhancement is 2.8 and 13.1 times for WVAR and Hy-WVAR respectively. These improvements are because of the usage of additional SRAM buffers in hybrid techniques. In other words, these SRAM buffers of hybrid techniques lessen the overall writes of STT-RAM buffers during high network traffic load. The newly proposed Inter-VNet policies: Dy-WVAR and Hy-Dy-WVAR are used to counter both Inter-VNet and Intra-VNet write variation and hence improve the lifetime more than the Intra-VNet policies (row 2, 3, 6 and 7). This improvement in the lifetime is due to the even distribution of writes across all the buffers. For example, the improvement in Dy-WVAR is 6.2 times in 4×4 and 4.49 times in 8×8 network over Intra-VNet policy WVAR.

4.5.4 Network Latency

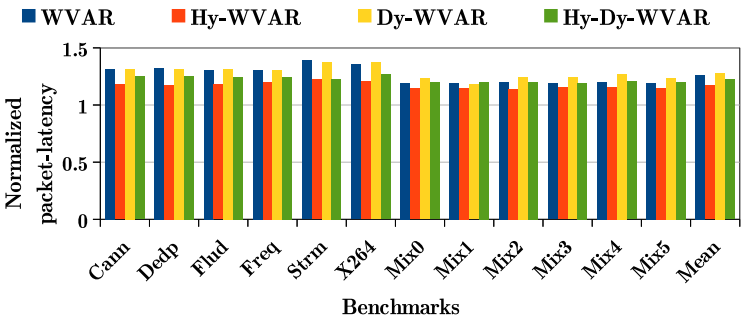


FIGURE 4.9: Normalized packet latency of proposed policies w.r.t. SRAM baseline in 4×4 mesh network (lower is better).

Figure 4.9 shows the latency graph of proposed techniques against SRAM baseline. The write cycles of STT-RAM-based VCs are 2 cycles. Using STT-RAM-based VCs, the network latency is increased by 23% in 4×4 network for WVAR. However,

Hy-WVAR attempts to reduce this and achieves network latency of 16% over SRAM, which is 6.24% improvement over WVAR.

The Inter-VNet policies further increase the latency as there is a stall introduced at the end of each interval to re-organise the VNet pointers. The average packet latency observed in Dy-WVAR policy is 27% over baseline SRAM. The Hy-Dy-WVAR policy brings it down to 22% w.r.t. SRAM, which is 4.35% improvement over Dy-WVAR. Thus, there is a trade-off between interval duration and an increase in packet latency. For shorter intervals, the stall will be introduced several times over the execution, increasing the packet delay. Whereas, longer intervals will alleviate this effect. A comparative analysis of the size of intervals on various metrics is done in a further section.

Network size	Reference policy	WVAR	Hy-WVAR	Dy-WVAR	Hy-Dy-WVAR	Row
4X4	SRAM	23	16	27	22	1
	WVAR	-	-6.24	2.48	-1.21	2
	Hy-WVAR	-	-	8.63	4.69	3
	Dy-WVAR	-	-	-	-4.35	4
8X8	SRAM	25	12	27	21	5
	WVAR	-	-12.5	1.8	-1.8	6
	Hy-WVAR	-	-	15.7	11	7
	Dy-WVAR	-	-	-	-3.7	8

TABLE 4.7: Percentage increase in packet latency comparison for all policies.

Table 4.7 shows a comparison of packet latency for all policies with reference policies for a 16-core and 64-core setup. Note that negative values shown in the table denotes the reduction in average packet latency. As we can see in the table the Hy-WVAR and Hy-Dy-WVAR policies show the least value in latency (row 2, 4, 6 and 8); whereas, Dy-WVAR policy shows the maximum increase (row 1, 3, 5 and 7).

4.5.5 Energy Analysis

The figure 4.10, shows the normalised total router energy of SRAM baseline and proposed policies for 4×4 mesh network, normalised w.r.t. SRAM baseline. As static energy is significantly reduced on account of STT-RAM buffers, we get overall energy savings. In particular, on average total energy is reduced by 56%

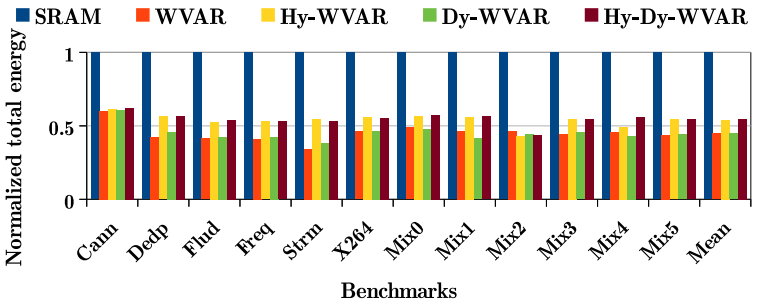


FIGURE 4.10: Normalized total energy of proposed policies w.r.t. SRAM baseline in 4×4 mesh network (lower is better).

and 46% in WVAR and Hy-WVAR respectively. The average reduction of total energy in Dy-WVAR and Hy-Dy-WVAR are 55.1% and 45.6% respectively.

The Table 4.8 shows the savings in total router energy for all policies compared with the reference policy in column-2. Note that the table contains the geometric mean of all simulations run on benchmarks mentioned in tables 4.3 and 4.4. In 16-core setup, WVAR and Dy-WVAR save 55.97% and 55.88% of total energy whereas Hy-WVAR and Hy-Dy-WVAR save 46.2% and 45.65% of total energy respectively. The total energy consumption increases in Hybrid policies due to the usages of SRAM based VCs. When we compare total energy w.r.t. WVAR policy, the energy overhead in HY-WVAR and Hy-Dy-WVAR has been observed as 22.19% and 23% whereas Dy-WVAR shows energy savings by -0.2% (table 4.8, row 3). Similar results are seen for 64-core setup.

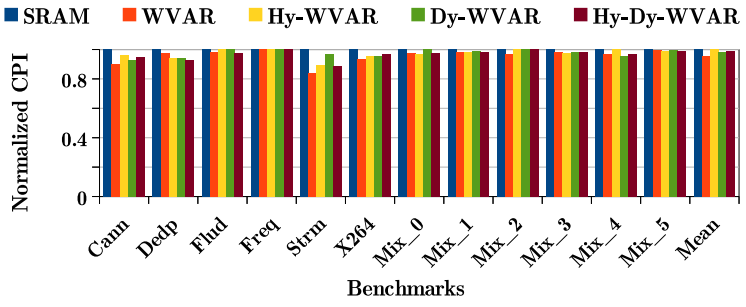


FIGURE 4.11: Normalized CPI for proposed policies in 4×4 mesh network (lower is better).

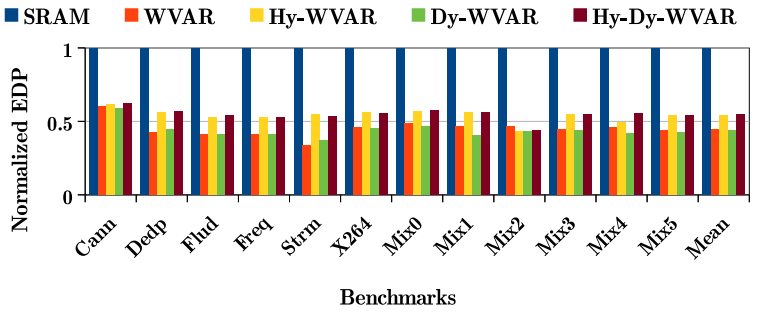


FIGURE 4.12: Normalized EDP gains of proposed policies over SRAM baseline in 4×4 mesh network (lower is better).

4.5.6 EDP Gain

The increase in network latency affects performance. In particular, we observe that CPI degrades by 1.5% in Intra-VNet policies and 4% in Inter-VNet policies, over baseline SRAM. Figure 4.11 shows normalized CPI. However, the savings in static energy are significant and thus leads to an improved EDP. Table 4.8 also shows the gains in EDP. This depicts the impact of the increase in packet latency and savings in static energy on the overall energy-delay-product of the system. On average, we get 55.99% and 55.33% EDP gain in WVAR and Dy-WVAR over baseline SRAM (Figure 4.12). Whereas, policies Hy-WVAR and Hy-Dy-WVAR show 46.80% and 45.87% gain. The detailed comparison with reference policies is given in table 4.8. Note that negative entry in the table represents the loss in EDP.

Network Size	Ref. Policy	Metric	WVAR	Hy-WVAR	Dy WVAR	Hy-Dy-WVAR	Row
4X4	SRAM	Energy	55.97	46.20	55.88	45.65	1
		EDP	55.99	46.80	55.33	45.87	2
	WVAR	Energy	-	-22.19	-0.20	-23.43	3
		EDP	-	-20.87	-1.50	-22.99	4
	Hy-WVAR	Energy	-	-	18.00	-1.017	5
		EDP	-	-	16.03	-1.76	6
	Dy-WVAR	Energy	-	-	-	-23.19	7
		EDP	-	-	-	-21.20	8
8X8	SRAM	Energy	58.48	53.35	57.26	47.46	9
		EDP	59.14	54.45	56.73	48.56	10
	WVAR	Energy	-	-12.35	-2.94	-26.55	11
		EDP	-	-11.47	-5.92	-25.89	12
	Hy-WVAR	Energy	-	-	8.37	-12.64	13
		EDP	-	-	5.00	-12.93	14
	Dy-WVAR	Energy	-	-	-	-22.93	15
		EDP	-	-	-	-18.85	16

TABLE 4.8: Router Energy Savings and EDP gains for all policies.

4.5.7 Analysing the effect of interval size

Interval	Inter-VNet Write-variation	Normalized Lifetime	Latency (in cycles)
Baseline	122.76	1	18.97
1M	0.25	19.97	24.20
2M	0.60	19.92	24.07
5M	1.35	19.80	24.07
10M	2.76	19.70	24.02

TABLE 4.9: Interval analysis for Dy-WVAR policy in 4×4 mesh network w.r.t. baseline.

The proposed Inter-VNet write variation policies perform the reallocation of buffer groups at the end of a given interval. Here, we analyse the effect of this interval duration on the various performance metrics. Change in interval affects the mapping of the buffer groups and thus affects the way the writes get distributed across groups/VNets. During every reallocation, the router pipeline is stalled for some cycles which affect the packet latency. Thus, if the interval duration is short, then the effect on latency is more compared to that in a longer interval. Table 4.9 shows the effect of different interval sizes on packet latency by applying the proposed policy Dy-WVAR. The table also shows that smaller intervals have better control over the Inter-VNet write variation (row 2). Write variation is very minimal for small intervals, and it increases slightly for longer intervals (row 4 and 5). A similar trend is seen in lifetime values. Smaller intervals have better control over variation and hence give a better lifetime improvement value compared to a longer interval.

4.5.8 Comparison with existing policy Hybrid-Drowsy [1]

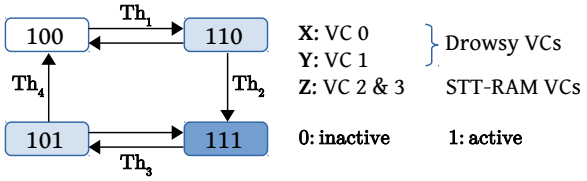


FIGURE 4.13: State transition diagram showing various power-on-off states of VCs [1].

The existing literature has contributions that use STT-RAM-based buffers in NoC; however, none of them has targeted the lifetime issue by reducing the write variation. The contribution in [1] uses a combination of SRAM VCs and STT-RAM VCs, termed as the hierarchical buffer architecture. Based on the traffic load, the VCs are power-gated and/or made drowsy. The proposal uses one SRAM VC that is always powered-on and another SRAM VC that can be kept in low power (drowsy) mode when not in use. Additional STT-RAM-based VCs are kept power gated in low traffic and are turned on during medium to heavy network traffic.

Figure 4.13 shows the state transition diagram for VCs with traffic load. Here, the VCs are divided into three levels. The first level (X) has an SRAM based virtual channel, $VC0$, which is active all the time. Whereas next level (Y) SRAM based virtual channel, $VC1$, switch between power-ON and drowsy state with traffic load. The third level (Z) virtual channels are STT-RAM-based and get activated only to accommodate heavy traffic load. The threshold values are formulated on buffer occupancy. Initially, the traffic load is low and only first level VC is active (figure 4.13, state `100`). When traffic exceeds Th_1 (more than Th_1 level of buffer occupancy), the second level VC gets activated (shown in figure 4.13, state `110`). The further increase of network load causes third level VCs to activate if the load is more than Th_2 (shown in figure 4.13, state `111`). The wake-up latency of STT-RAM is much higher than drowsy SRAM VCs and hence when network traffic reduces to medium load (less than Th_3) from high load state, the drowsy SRAM VCs gets deactivated (figure 4.13, state `101`). Also VC state transition happens from `101` to `111` when current state is 101 and network traffic exceeds Th_3. When network traffic decreases further (gets lower than Th_4), the STT-RAM VCs gets power-gated (state `100`).

Policy	Intra-VNet Write Var	Inter-VNet Write Var	Lifetime	Energy savings (%)	EDP gain (%)	Latency increase (%)
Hy-drowsy	131 %	286 %	17.9	27.83	27.64	10.2
Dy-WVAR	0.00025 %	0.53 %	19.92	57.18	57.43	27

TABLE 4.10: Hy-drowsy policy details with base comparison in 4×4 mesh network.

This contribution aims to reduce the power consumption of the NoC routers. The difference with our proposed methods is that our methods try to reduce write variation and hide the latency by using only one additional SRAM VC, and at the same time save power consumption by using maximum STT-RAM-based VCs. The comparison of the Hy-Drowsy and Dy-WVAR policies with the baseline is shown in table 4.10. Compared to Hy-Drowsy, except latency; the Dy-WVAR performs better in each and every metrics with respect to baseline, which is shown in row 2 of table 4.10. This shows the effectiveness of our proposed technique: Dy-WVAR. The Hy-Drowsy policy shows an average of 131% of write variation across VCs in virtual networks and 286% of write variation across buffer groups. This impacts the lifetime of STT-RAM-based buffers. Since the policy uses two SRAM base VCs per virtual network, the average lifetime of a buffer is 17.9 times with respect to STT-Base, which is little less than our proposed policy Hy-WVAR which uses only one SRAM based VC per virtual network. Uses of SRAM VCs balances the latency increase caused by STT-RAM-based VCs. Hence, the latency increase is less with respect to proposed policies. The uses of two drowsy SRAM based VCs balance static energy and hence EDP. The policy shows 27.83% savings in total energy and 27.64% EDP gain with respect to *SRAM* baseline.

Note that the hybrid-buffer design of [1] has an additional overhead to wake up drowsy VCs. There is also no implicit support for improving the lifetime of buffers. Thus, our proposed policies not only reduce energy consumption but also improve the lifetime of STT-RAM-based NoC buffers.

4.5.9 Analysis using Synthetic Workloads

To study the impact of proposed policies on the packet latency, we performed simulations on a 256-core, i.e. 16×16 mesh network using synthetic traffic patterns [175]. We analyse the effect on packet latency by varying the injection rates. We also show the performance of our proposed policies by scenarios where the VCs per VNet are different as well as cases when the buffer depth for VCs carrying data packets is changed.

4.5.9.1 Latency Analysis

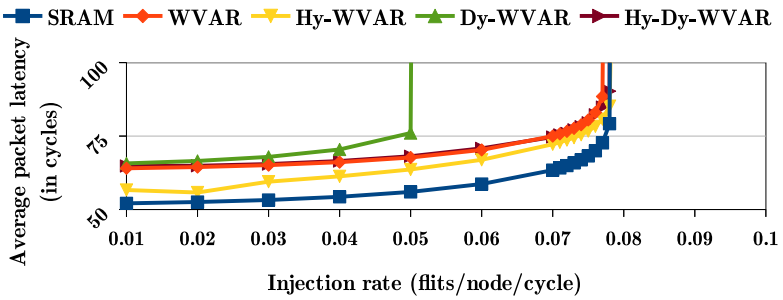


FIGURE 4.14: Performance Comparison under Uniform-Random Synthetic Workloads in 16×16 mesh network.

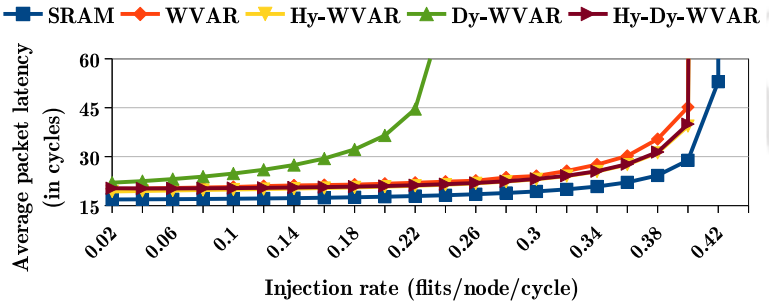


FIGURE 4.15: Performance Comparison under Nearest-Neighbour Synthetic Workloads in 16×16 mesh network.

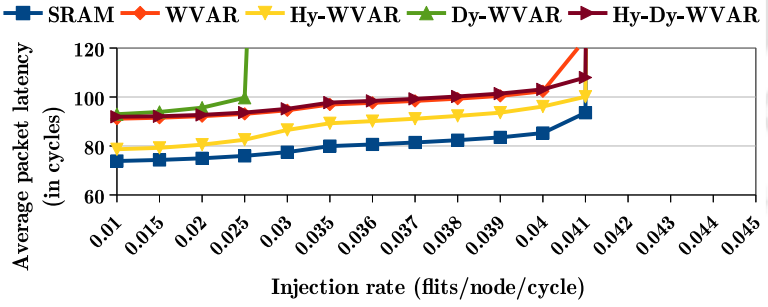


FIGURE 4.16: Performance Comparison under Bit-Compliment Synthetic Workloads in 16×16 mesh network.

Figure 4.14 shows performance comparison of proposed policies with the SRAM baseline under *uniform-random* synthetic workload. On account of the slow write speed of STT-RAM, the packets take slightly more time to reach the destinations. As expected, the latency of Hy-WVAR policy lies between the latency values for SRAM and WVAR, and both the proposed policies reach saturation earlier

than the baseline SRAM. WVAR increases latency by 22% on average, whereas Hy-WVAR shows latency improvement by 9.4% w.r t. WVAR. Dynamic policies increase latency by 29% and 22% respectively.

In *neighbour* synthetic workload, figure 4.15, WVAR, and Dy-WVAR performance degrades and latency grows very high, which is 20% and 28% over SRAM. The hybrid policies, Hy-WVAR and Hy-Dy-WVAR, shows 16% and 19% more latency with respect to SRAM.

The figure 4.16 shows performance comparison of proposed policies under *bit-compliment* traffic pattern. As expected, the WVAR and Dy-WVAR increase latency by 23% and 26% respectively and Hybrid policies, Hy-WVAR and Hy-Dy-WVAR, brings it down to 7% and 20% respectively.

Note that as Dy-WVAR stalls the router pipeline at the end of each interval, this policy saturates slightly before the other policies.

4.5.9.2 Effect of Threshold value

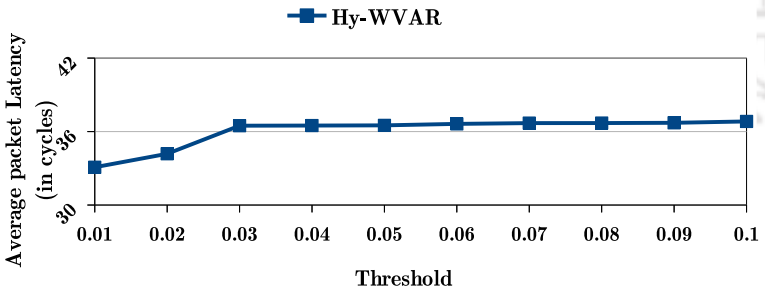


FIGURE 4.17: Threshold analysis for Hy-WVAR policy with Synthetic Workloads in 8×8 mesh network.

The usage of SRAM based virtual channel in Hy-WVAR and Hy-Dy-WVAR policies depends on the threshold used to decide traffic load. Lower threshold value ensures more frequent writes in SRAM based VC. Figures 4.17 and 4.18 shows the effect on network latency using different threshold values under uniform-random traffic pattern. Higher threshold reduces the number of writes in an SRAM based virtual channel which results in more average packet latency. Thus, there is a

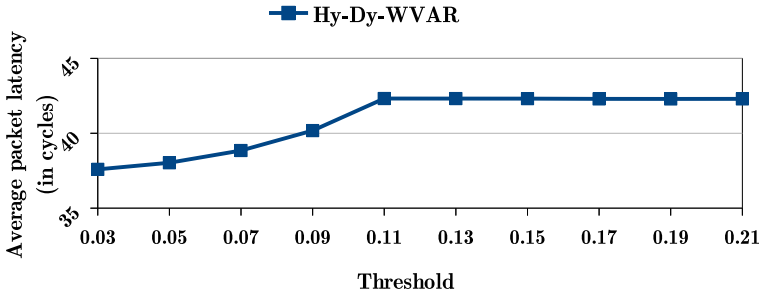


FIGURE 4.18: Threshold analysis for Hy-Dy-WVAR policy with Synthetic Workloads in 8×8 mesh network.

trade-off between packet latency and power. If the threshold is less, latency is less, but it incurs more leakage power due to the use of SRAM. On the other hand, a higher threshold saves leakage but incurs more packet delays.

4.5.9.3 Effect by varying VCs-per-VNet and Buffer capacity

Our policies perform consistently, even when the VCs per VNet is changed. They also perform consistently when the buffer depths assigned to data VNets are varied. The results are shown in the table 4.11 and 4.12.

Policy	VC per VNet	Intra-VNet Write Var reduction	Inter-VNet Write Var reduction	Lifetime improvment (in times)	Latency increase (%)	Row
WVAR	4	99.99	5.18	2.37	21.73	1
	6	99.99	4.12	3.56	21.68	2
	8	99.99	3.61	4.70	21.63	3
Hy-WVAR	4	99.99	-9.08	4.86	12.84	4
	6	99.99	19.64	7.28	12.92	5
	8	99.99	31.29	9.70	12.95	6
Dy-WVAR	4	99.99	97.88	2.54	29.67	7
	6	99.99	98.71	3.81	28.89	8
	8	99.99	99.91	5.10	28.85	9
Hy-Dy-WVAR	4	99.99	99.44	7.01	22.42	10
	6	99.99	98.95	10.48	22.36	11
	8	99.99	98.73	12.45	22.32	12

TABLE 4.11: Performance analysis of proposed policies for different VC-per-VNet values in 16×16 mesh network.

Policy	Data VC depth	Intra-VNet Write Var reduction	Inter-VNet Write Var reduction	Lifetime improvement (in times)	Latency increase (%)	Row
WVAR	2	99.99	7.25	1.8	21.88	1
	4	99.99	5.18	2.37	21.73	2
	6	99.99	3.09	2.96	21.52	3
	8	99.99	3.09	2.96	21.52	4
Hy-WVAR	2	99.99	-29.07	3.37	13.78	5
	4	99.99	-9.08	4.86	12.84	6
	6	99.99	-3.96	4.9	12.17	7
	8	99.99	-3.96	4.9	12.17	8
Dy-WVAR	2	99.99	97.45	2.48	24.03	9
	4	99.99	97.88	2.54	29.67	10
	6	99.99	97.99	3.4	31.17	11
	8	99.99	97.99	4.24	31.17	12
Hy-Dy-WVAR	2	99.99	99.58	5.6	22.36	13
	4	99.99	99.44	7.01	22.42	14
	6	99.99	99.35	8.69	22.19	15
	8	99.99	99.35	10.86	22.19	16

TABLE 4.12: Performance analysis of proposed policies for different data VC depth in 16×16 mesh network.

4.6 Storage Overhead Analysis

The counter for each virtual channel can be reset when any one of the counters in a virtual network reaches its maximum value. This multiple times counter-reset allows us to use small counter such as $2B$. Therefore storage overhead for each virtual channel will be $2B$, which is a total of $24B$ for given system parameters in table 4.2. The number of buffers at any input port is 24; hence the total size will be $384B$. The counter storage overhead in WVAR and Hy-WVAR policy is 6.25% w.r.t. SRAM baseline design.

The area occupied by STT-RAM is one-third of the area consumed by SRAM memory cells. Hence, overall the buffer area, including counters, is reduced by 60.42% per port when we replace SRAM with WVAR based designs. In Hy-WVAR design counter overhead is same as WVAR design whereas total area is reduced by 35.42% with respect to SRAM baseline design.

The Inter-VNet policies use additional counters and swap space with the buffer groups. These make the overall storage overhead to 26%. In this case, the overall reduction in the total area over SRAM design is 51.56% and 31.25% respectively for Dy-WVAR and Hy-Dy-WVAR. Note that all counters are SRAM based.

4.7 Summary

NoC is the backbone of communication for today's chip multiprocessors and also a significant contributor to the power budget. Specifically, the buffers at the ports consume more power in terms of leakage. This chapter proposed to use NVM based STT-RAM buffers as they promise high density and low leakage. The major drawbacks of STT-RAM are their low write endurance which can affect the lifetime of the buffers. Towards this, the work proposed policies to reduce write variation across buffers.

The proposed policies: Dy-WVAR and Hy-Dy-WVAR aim to further enhance the lifetime of the STT-RAM buffers by distributing the writes in buffers across VNets: Inter-VNet write variation. To control the variation across VNets, we proposed to create buffer groups which were dynamically assigned to the VNets at runtime depending on their write counts. In particular, VNets incurring more write were subsequently assigned to buffer groups incurring lesser writes during an interval. These policies reduce write variation to almost 0% and improve the lifetime by 19.92 and 55.5 times over baseline STT-RAM design. The pure STT-RAM design Dy-WVAR increases latency by 27% over SRAM, which is rectified by the Hy-Dy-WVAR design, which makes the latency 22% over SRAM.

This work has thus demonstrated that careful management of writes in NVM buffers can make them a viable replacement for power-hungry SRAM buffers.

Chapter 5

Power saving using Frequency Scaling and Power Gating

Whenever writes are incurred by the buffers, our previous two chapters distributed them evenly across the buffers, thus improving the overall lifetime of the buffers. Along with lifetime enhancement, our supplementary aim is to save energy. In this chapter, we investigate various methods in which energy can be saved, which include frequency scaling and/or power gating less used router components. In the context of dark silicon, most of the processing elements (PEs) will be powered off; however, the network is expected to be available. To keep the network on and at the same time save energy, we propose to scale the frequency of the routers (using only SRAM based buffers) attached to powered-off PEs. Frequency scaled routers will be slow in transmission; however, as the traffic is expected to be lesser, this speed degradation does not have a significant impact on the performance. If we use all STT-RAM buffers, then frequency scaling is not required as STT-RAM has negligible leakage. However, to overcome the latency degradation due to slower writes in NVM, we also use SRAM based VCs forming a hybrid combination. If the PE is powered-off (on), then the SRAM VCs are powered off (on). Other proposals include giving priority to packets carrying critical words, which are sent via faster SRAM VCs irrespective of the PEs power-on status.

5.1 Introduction

In order to control on-chip leakage power, among others, dark silicon has emerged as a promising paradigm. The architecture community now has to address new challenges for the design and management of processors in the dark silicon era. The on-chip communication infrastructure (Network on-Chip-NoC) should be able to cope with connecting nodes operating at different frequencies: powered-on, powered-off, voltage scaled, etc. This diverse requirement has put the NoC researchers to invest in designing routers that are low power and, at the same time, do not compromise on throughput. In the presence of dark silicon, the processing cores are turned off when not in use; however, the on-chip interconnect is expected to be available to maintain the connectivity. In other words, the routers connected to powered down processing elements are kept on.

Buffers consume maximum static power, and static power dominates dynamic power at smaller technology nodes [176]. As discussed in Chapter 2, to reduce the standby power of buffers, Non-Volatile Memory (NVM) technologies can be used as an alternative for router buffer. In this work, we exploit the fact that buffers consume maximum power and attempt to save power at the level of buffers. We keep the routers always powered-ON to keep constant connectivity in the dark silicon. To further save leakage power, in this chapter, we propose five policies:

- gBUF-DN: We use all SRAM buffers in the router and keep most of the buffers powered-gated for the nodes with processing elements powered off.
- SRAM-FRQSCL: We use all SRAM buffers in the router and lower down the frequency of buffers for the associated powered OFF processing element.
- Hy-SEL-ON: Here, we have used hybrid VCs that incorporate a large number of NVM VCs and a small number of SRAM based VCs at each port of router buffer. We keep a certain number of SRAM and STT-RAM router buffers ON based on the status of the processing element.

- Hy-NEIG-SEL-ON: This extends the above policy. The buffers of neighbouring routers having a connection with this router are selectively power gated.
- Hy-CRIT-WRD: We attempt to hide the latency increase due to NVM buffers by optimizing the transfer of critical words through the network.

Thus by proposing various design alternatives for buffer management, we investigate their applicability towards supporting dark silicon-based CMPs.The objective is to reduce overall energy consumption and, at the same time, maintain connectivity with low latency overheads.

There are few works done to improve energy saving for SRAM buffer based designs, which are based on frequency scaling, router power gating or buffer power gating (c.f. section 2.4). A few of the existing techniques show a significant amount of energy-saving; however, it suffers from wake-up energy overhead, delay and performance loss. The NVM buffer has nearly zero leakage power consumption. However, it suffers from costly write operations and weaker write endurance. Implementing these techniques with NVM based buffers will result in the adverse effect of router wake-up overhead, as the writes in NVM consumes more cycles than writes in SRAM. Also, NVM buffer power gating will not result in significant leakage energy saving since it has very less leakage power consumption. In addition to this, the existing approaches have not taken weaker write endurance problem of NVM buffer based designs in to account. Thus, the chapter proposes different longevity approaches to overcome the NVM challenges.

The chapter is organized as follows: Background and Motivation are reported in section 5.2. Proposed techniques are discussed in section 5.3. Section 5.4 illustrates the experimental methodology and results and analysis are done in section 5.4.3. Finally, we conclude this chapter in section 5.6.

5.2 Background and Motivation

5.2.1 NoC Router Traffic

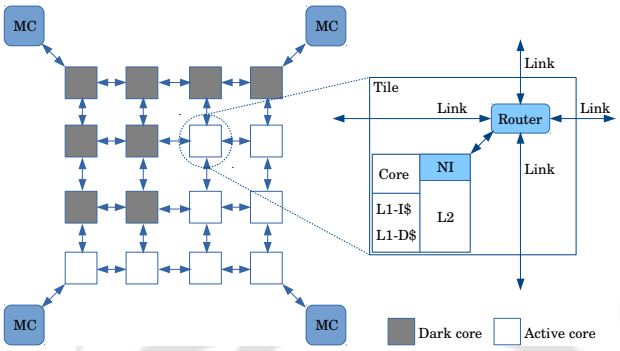


FIGURE 5.1: Router architecture showing power gated buffers at input ports

The Chip MultProcessor (CMP) setup used in this work is a Tiled Chip Multi-Processor (TCMP). This has evolved as a scalable design for small-scale future CMPs. A TCMP, shown in figure 5.1, consists of processing cores arranged in a mesh network of nodes connected by an on-chip interconnect called the network on chip (NoC). Each tile consists of a core, its private instruction, and data L1 caches and a slice/bank of shared last level L2 cache. The L2 cache banks in each tile together form the LLC arranged as a NUCA architecture.

In the context of dark silicon, most of the processing elements (PE), i.e., tiles are turned off. If PE is OFF, then the router will encounter very less flow of traffic, and its role is mainly to maintain connectivity and allow passage of packets flowing through it. Figure 5.2 shows the percentage of local data packets passing through the router when the associated PE is ON. In case this PE is power gated, then the local data packets passing through its router will be absent. Figure 5.3 shows router traffic reduction in 25% dark nodes scenario. The routers selected have the associated PE powered off. (In particular, we plot the traffic reduction for routers R1 and R14 in Pattern1 and R7, R10, and R11 in Pattern2, where the patterns are shown in figure 5.12.)

The chip multiple processor setup used in this work is a Tiled Chip MultiProcessor (TCMPs). This has evolved as a scalable design for small-scale future CMPs.

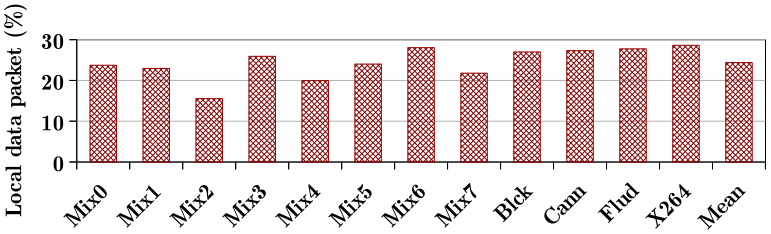


FIGURE 5.2: Percentage of local-only data packets passing through a router.

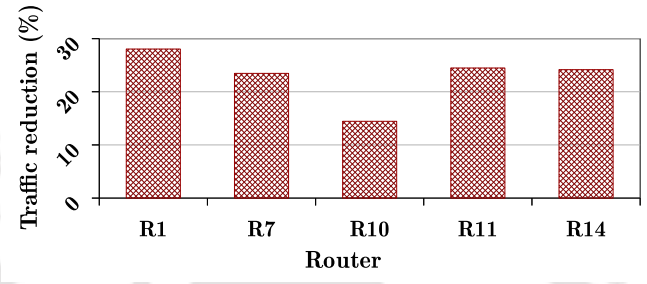


FIGURE 5.3: Traffic reduction in routers associated with power gated PEs in 25% dark scenario.

A TCMP, shown in figure 5.1, consists of processing cores arranged in a mesh network of nodes connected by an on-chip interconnect called the network on chip (NoC). Each tile consists of a core, its private instruction, and data L1 caches and a slice/bank of shared last level L2 cache. The L2 cache banks in each tile together form the LLC arranged as a NUCA architecture. In the context of dark silicon, most of the processing elements (PE), i.e., tiles are turned off. If PE is OFF, then the router will encounter very less flow of traffic, and its role is mainly to maintain connectivity and allow passage of packets flowing through it (c.f. Figure 5.2 and 5.3). Figure 5.2 shows the percentage of local data packets passing through the router when the associated PE is ON. In case this PE is power gated, then the local data packets passing through its router will be absent. Figure 5.3 shows router traffic reduction in 25% dark nodes scenario. The routers selected have the associated PE powered off. In particular, we plot the traffic reduction for routers R1 and R14 in Pattern1 and R7, R10, and R11 in Pattern2, where the patterns are shown in figure 5.12.

The write variation and lifetime is defined in Section 2.3.2.

5.3 Proposed Policies

In this section, we describe our five proposals. The main objective is to save on the buffer leakage energy if the attached PE is power gated. The hypothesis is that if the PE is powered gated, then the traffic flowing through the routers attached to such PEs will be lesser than normal traffic. Hence there is an opportunity to save leakage energy by either powering off some SRAM based VCs or using low-leakage STT-RAM-based VCs or by using frequency scaling. We have investigated all the approaches in the proposed design alternatives.

5.3.1 Gated Buffers of Dark Nodes: gBUF-DN

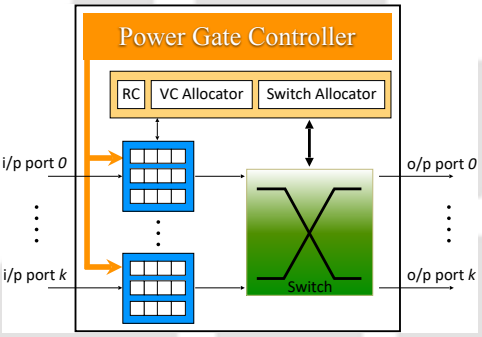


FIGURE 5.4: Router architecture showing power gated buffers at input ports

The proposal reduces the leakage energy of the router buffer. Towards achieving this, we discuss the design of NoC routers having gated buffers. This section briefly illustrates *gBUF-DN*.

Figure 5.1, shows some nodes in grey colour, indicating that they are gated/powered-off. The set of nodes that are gated at a particular time instance change as per the application scheduling and thermal stability of the on-chip components. To support the era of dark silicon, we propose to power gate maximum number of power-hungry router buffers for the powered off processing elements. Whereas, to maintain connectivity, we kept a minimum number of buffers powered on at each VNet. We call this proposal as gated BUFfer at Dark Nodes: gBUF-DN.

Figure 5.4 presents the router design that implements the power gating for selective buffers in the powered off processing element. In this implementation, the Power Gating (*PG*) signal is raised when the tile/processing element is turned OFF. On receiving the *PG* signal, the router makes most of the buffers power off, as shown in the grey colour in the figure. Whereas, to maintain connectivity, some of the buffers are kept active and is shown in white colour. Once the processing core is turned ON, the *PG* signal gets disabled, and all the buffers are kept active in full power mode. Hence, as long as the buffers are inactive, the leakage power is saved.

NoC uses credit-based flow control that informs the upstream router about the availability of free VCs. When a router has power gated VCs, it will not send the credit for those particular VCs. This way, the upstream router will not allocate packet to the power gated VCs. Before power gating, the router informs upstream router by sending zero credits for the particular VCs. Additionally, the router waits for a few cycles to receive any packets that may be already on the way. This avoids packet loss.

The policy: gBUF-DN uses conventional SRAM buffers in the routers. The other alternative to save leakage power is to make use of near-zero standby power NVM buffers, i.e., STT-RAM. Although, the proposed work can easily be implemented using other NVM based buffers such that PCRAM or ReRAM.

5.3.2 SRAM-FRQSCL

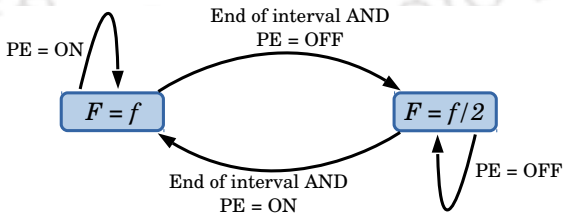


FIGURE 5.5: FSM of the router frequency scaling in SRAM-FRQSCL.

In this proposal, we assume that the buffers (i.e., VCs) in all routers are made using SRAM technology. In order to save leakage energy in the dark silicon scenario, we

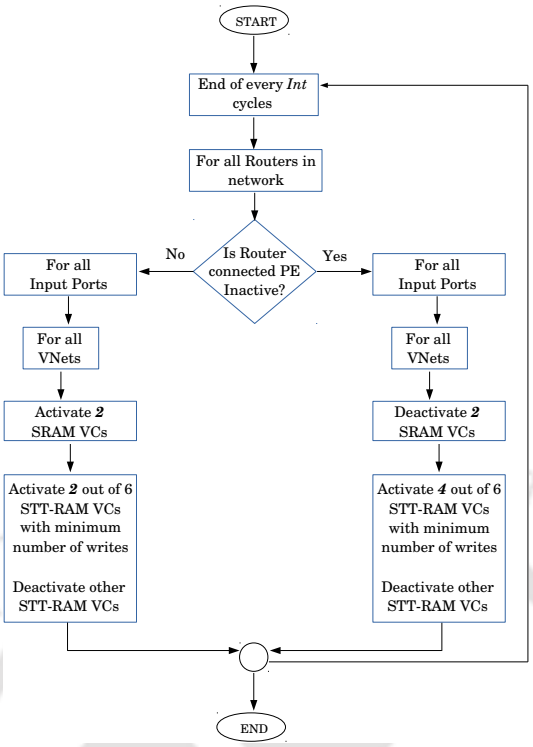


FIGURE 5.6: Flowchart for proposal Hy-SEL-ON. Assuming 4 VCs are active at a time.

propose to operate the router (including the buffers) at a lower frequency. This frequency scaling is done for routers connected to powered-off PEs.

Initially, the application runs with all routers at maximum frequency. Subsequently, at regular intervals, the routers undergo frequency scaling if the PE is off. Once the frequency is scaled, it remains so throughout the interval and can be changed only at the end of the interval (if the PE is turned-on in the meantime). Figure 5.5 shows the FSM depicting the behaviour of the router and frequency scaling. For our experimental evaluation, we propose to reduce the frequency to half of the maximum frequency.

5.3.3 Hy-SEL-ON

The first proposal uses all SRAM based buffers (or VCs) and reduced frequency to save power. To achieve the same objective, in the second idea, we propose to replace a few SRAM based VCs with STT-RAM-based VCs, as STT-RAM has very

little leakage. In this case, if the attached PE is powered-off, then all the SRAM based VCs will be powered-off, and the traffic (which is expected to be less) will be handled by the STT-RAM VCs. The mechanism to inform the upstream router about power gating is the same as discussed in section 5.3.1. Figure 5.6 shows the flowchart of the proposal. The procedure is invoked at regular intervals, and the decision taken at the beginning of the interval is maintained throughout the interval. As STT-RAM is dense, we can accommodate up to 4 times capacity in the same area as SRAM [167]. We use this fact and employ (conservatively) three times the number of SRAM VCs considering area overhead occupied by peripheral circuitry and our proposal based counters. For example, if we had 4 SRAM VCs in the original design, we keep 2 VCs as SRAM and replace the other 2 SRAM VCs with 6 STT-RAM VCs (to fit in the same area taken by the 2 SRAM VCs). However, we only use 4 VCs in total, i.e., when SRAM is used, then we need to select 2 STT-RAM VCs from the 6 STT-RAM VCs. Note that in all baseline policies, 4 SRAM VCs are used for simulation purposes.

STT-RAM, although being dense and leakage friendly, it suffers from low write endurance. The buffers which are frequently written get worn-out faster compared to less written buffers. On account of density, we have more STT-RAM VCs than required. Therefore during implementation of our proposal, we select m out of n VCs of type STT-RAM. However, if we select the same VCs each time, the buffers in those VCs will incur breakdown faster compared to others. In order to avoid this, we propose a write-variation aware VC selection among the STT-RAM VCs. In that, we maintain the write counts for each VC, and at the beginning of the next interval, use those STT-RAM VCs that have the least write count. For this, the counter of size 2 Bytes is maintained with each VC. This helps in distributing the writes evenly across all STT-RAM VCs (buffers) and improves the lifetime.

Figure 5.10 shows an example scenario. In the beginning, all SRAM VCs and 2 STT-RAM VCs are operational. Here, the VCs in use are VC0, VC1, VC2, VC4. At the end of the interval, if the PE is still powered-ON, then the SRAM VCs are used, and we need to select STT-RAM VCs depending on their write counts. In this example, the VCs in use are VC0, VC1, VC5, VC6. The system is run with

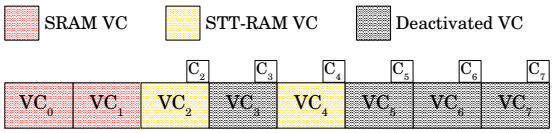


FIGURE 5.7: Int_i

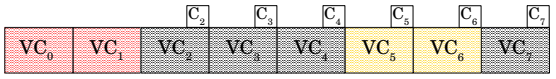


FIGURE 5.8: Int_{i+1}

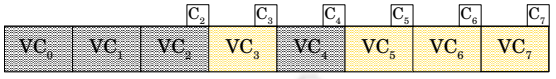


FIGURE 5.9: Int_{i+1}

FIGURE 5.10: VC power status for different time intervals. (a) shows initial VC status for the interval Int_i. (b) shows VC status for interval Int_{i+1} when the associated node to router is powered ON and (c) shows VC power status for interval Int_{i+1} when the associate node is powered OFF.

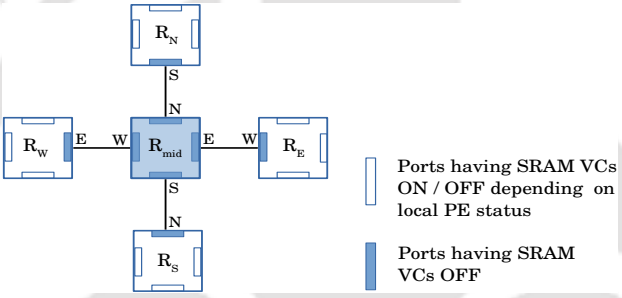


FIGURE 5.11: Illustration on policy Hy-NEIG-SEL-ON showing powering OFF of SRAM VCs in immediate neighbour.

this setup for another interval. Suppose at the next interval the PE gets powered-off, then we power off the SRAM VCs (VC0, VC1), and we need to turn-ON total 4 STT-RAM VCs having the least number of write counts. In this example, the VCs selected are VC3, VC5, VC6, VC7.

5.3.4 Hy-NEIG-SEL-ON

The idea here is an extension to Hy-SEL-ON above. For the routers that have PE powered-off, there is expected to be very less traffic passing through them. This is also true for the immediate neighbours' connected ports of such nodes. For instance, as shown in figure 5.11, if the central router has powered-off PE, then the VCs at input ports have STT-RAM VCs being used. As the amount of traffic

passing through this router will be less, we propose to save the leakage of the neighbouring routers by making their ports connected to this router behave similarly to the input port VC of the powered OFF node. In particular, in figure 5.11, the port-E of R_W will use only STT-RAM VCs and power-off SRAM VCs. Same is the case with port-S of R_N, N of R_S, and W of R_E. This helps in saving leakage energy further. Note that some of STT VCs will always be ON, and only SRAM VCs will be powered ON or OFF based on connected node status.

5.3.5 Hy-CRIT-WRD

The idea of using STT-RAM buffers for routers associated with active nodes increases network latency since the writes in STT-RAM takes more time than SRAM based buffers (cf. Table 5.1). This causes an increase in network latency over SRAM based buffer designs. It has been experimentally observed that the use of pure STT-RAM buffers increases the latency by 22% over baseline SRAM design in a 4×4 mesh network. The delay caused due to the usage of STT-RAM buffers slows down all kinds of packets in the network. In case the system requires to send critical words faster then we can use our proposed Hy-CRIT-WRD. Because the delay in critical words may impact the overall performance of the system.

The main idea is to use SRAM based VCs for critical words even when there are only STT-RAM VCs active. At the time of VC allocation, critical word gets assigned to SRAM based VC. Note that the LLC controller will forward words indicating that they are critical to the NI. Further, NI embeds this information into the header flit while injecting words into the network. This information is used during the VC allocation stage at every router. At each VC allocation stage, at every router, this information is used to select VCs. For critical words, SRAM-based VC is allocated to the packet.

For the routers which are connected with active nodes, the SRAM VCs are already active; hence the VC allocator allocates critical words to it. Whereas for the routers with only STT-RAM VCs active, an SRAM VC is woken up and kept

active for a predefined number of cycles. This reduces the network latency for the critical words, which reduces overall network latency and improves system performance for the critical word optimization. Also, note that the remaining cache line (whose critical word has already been sent) travels as a separate packet through any available VC.

5.4 Experimental Evaluation

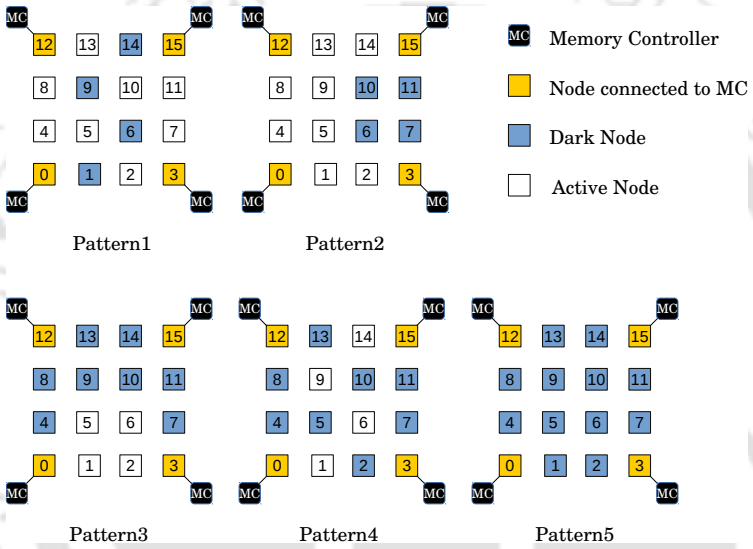


FIGURE 5.12: Dark node patterns. Fig. (Pattern1) and (Pattern2) shows pattern for 25% nodes as dark. Fig. (Pattern3) and (Pattern4) shows pattern for 50% nodes as dark. Fig. (Pattern5) shows pattern for 75% nodes as dark.

Parameter	**SRAM (F = *f*)**	**SRAM (F = *f/2*)**	**STT-RAM**
write latency (cycle)	1	2	2
read latency (cycle)	1	2	1
read energy/bit (pJ)	0.026	0.0128	0.022
write energy/bit (pJ)	0.0363	0.0175	0.182
leakage power (mW)	0.206	0.144	0.067

TABLE 5.1: SRAM and STT-RAM buffer configuration

We used Gem5 [168], a multi-core full system simulator with GARNET [169], as the interconnection network model. We use DSENT [170] integrated with Gem5 and GARNET to measure the static and dynamic energy consumed at $45nm$ process technology. We evaluate a 16-core system with tiles arranged in a 4×4 mesh network. We use CACTI-STT [171] and NVSim [172] to get SRAM and STT-RAM latency, read-write energy, and leakage power. Table 5.1 shows all

Parameter	**Details**
core count	16, 2GHz
topology	4 × 4 mesh
L1 I & D cache , L2 cache (shared)	64KB, 64KB , 16MB (64B block size)
router pipeline	2 stage, 16B flit
Packet size	control 1 flit, data 5 flits
virtual network count	3 per port with 4 VCs per VNet
virtual channel depth	1 buffer for control, 4 buffers for data VC
counter size	2B
topology	Mesh-XY
routing algorithm	XY
synthetic traffic pattern	Uniform random, Bit complement
SRAM wake-up time	5 cycles [2]

TABLE 5.2: System and interconnect configuration

Parsec Benchmarks
blackscholes (Blck), bodytrack (Body), canneal (Cann), fluidanimate (Flud), x264 (X264)
SPEC Benchmarks
mcf (mix0), zeusmp (mix1) lbm (mix2), cactusADM (mix3), bwaves (mix4), gamess (mix5), leslie3d (mix6), gromacs (mix7)

TABLE 5.3: Benchmark acronym

the obtained timing and energy parameters. Each core/tile is attached to one NoC router. Each core has a private L1 cache and a slice of shared L2 cache, which uses the MESI cache coherence protocol. The details of the configurations for the system and network are given in table 5.2. To bring the effect of dark silicon, we keep some cores powered-off. Note that the experiments have been carried out by keeping 25%, 50%, and 75% of nodes as dark. The cores that are powered-on will run applications from the scheduled benchmark. We evaluate the network performance of these applications in their corresponding regions of interest (ROIs). The work is evaluated using Multi-Programmed (MP) benchmark SPEC CPU2006 [173] and Multi-Threaded (MT) PARSEC benchmarks suite [177]. The acronym used for benchmarks is given in the table 5.3. From the list of all benchmarks, we ran 12 instances of each application for 25% dark node patterns, eight instances of each application for 50% dark node patterns, and four instances of each application for 75% dark node pattern on 16 cores mesh network. We warm up each SPEC multi-programmed workload for 1 billion instructions and run for 100 million instructions. Synthetic workloads on different traffic patterns are experimented by varying the injection rates on 8 × 8 network.

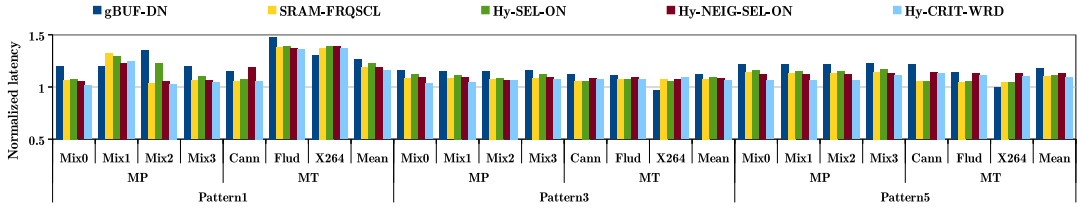


FIGURE 5.13: Normalized packet latency of proposed policies with respect to SRAM baseline in 4×4 mesh network (lower is better).

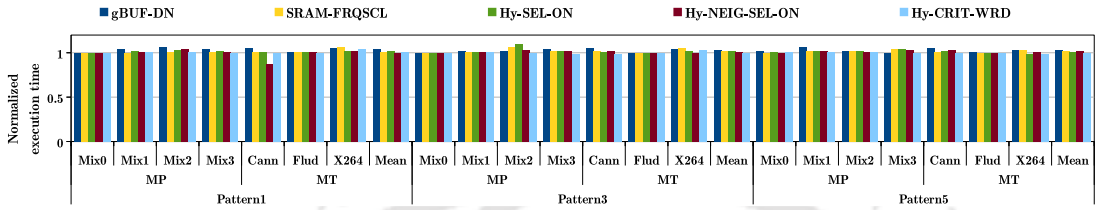


FIGURE 5.14: Normalized execution time for proposed policies in 4×4 mesh network (lower is better)

5.4.1 Evaluation Metrics

We show results for (1) network latency (2) execution time (3) total energy (4) power consumption (5) EDP (Energy Delay Product) gain (6) write variation (calculated by using equation 2.1 and 2.2) and (7) lifetime. The results are given for baseline policy using basic VC allocation policy (cf. section 2.1) and the proposed policies: *SRAM-FRQSCL*, *Hy-SEL-ON*, *Hy-NEIG-SEL-ON* and *Hy-CRIT-WRD*. Reduction in total energy, EDP gain, and network latency comparison are shown with respect to the baseline (*SRAM*), using pure SRAM buffers. Improvements in the lifetime and write variation are shown with respect to the STT-RAM baseline, (*STT-Base*), using STT-RAM buffers having the same number of VCs per VNet as the SRAM baseline technique. We compare our work with *gBUF-DN*, BlackOut [2] (*BlackOut*) and Hy-Drowsy [1] (*Hy-Drowsy*) which is shown in 5.4.4. We also compare proposed policies with router parking [3] for synthetic work traffic (cf. section 5.4.4.2).

5.4.2 Pattern for Dark nodes

The application mapping to cores on the chip is mainly done to optimize the network, minimize congestion, improve system throughput, etc. while maintaining TDP of the chip [178]. It shows the impact of contiguous and spatially distributed mapping in dark silicon. Hence, we consider patterns where applications are mapped contiguously and spatially on the chip for our simulations. Figure 5.12 shows dark silicon patterns keeping 25%, 50% and 75% of nodes dark on the chip. Note that the corner nodes are connected with the memory controller, and hence the associated routers are kept active in all the policies. The Pattern1 and Pattern2 have 25% dark nodes. The pattern in (1) has spatial application mapping; whereas, (2) shows application mapping in a region. Similarly, Pattern3 and Pattern4 show the pattern for 50% nodes as dark. The pattern Pattern5 in figure 5.12 has 75% nodes as dark, and hence only corner nodes have application mapped to them.

5.4.3 Results and Analysis

5.4.3.1 Network Latency

Pattern	Reference Policy	gBUF-DN	SRAM-FRQSCL	Hy-SEL-ON	Hy-NEIG-SEL-ON	Hy-CRIT-WRD	Row
Pattern2	SRAM	14.6	4.3	7.8	7.9	4	1
	gBUF-DN	-	-8.9	-5.8	-5.8	-9.3	2
	SRAM-FRQSCL	-	-	3.5	3.4	-0.3	3
	Hy-SEL-ON	-	-	-	-0.04	-3.7	4
	Hy-NEIG-SEL-ON	-	-	-	-	-0.3.6	5
Pattern4	SRAM	13	8	9.3	9.7	7.3	6
	gBUF-DN	-	-4.4	-3.3	-3	-5	7
	SRAM-FRQSCL	-	-	1.2	1.5	-0.7	8
	Hy-SEL-ON	-	-	-	0.3	-2.2	9
	Hy-NEIG-SEL-ON	-	-	-	-	-2	10

TABLE 5.4: Network latency comparison (in percentage) for all policies in 4×4 mesh network.

Figure 5.13 shows the latency graph of proposed policies for all the dark node patterns discussed in section 5.4.2. The proposed policy SRAM-FRQSCL reduces the frequency of a router to save energy when the associated node is dark. This reduces network latency performance by 15% over SRAM design in Pattern5.

Whereas, the STT-RAM-based policies increase packet latency due to the longer write cycles of the STT-RAM VCs (cf. Table 5.1). The usage of STT-RAM VCs increases the network latency by 10% and 11% in 4×4 network for Hy-SEL-ON and Hy-NEIG-SEL-ON policy. However, Hy-CRIT-WRD policy attempts to reduce this by using SRAM VC for critical words/packets and improves by 4% and brings it to 7% over SRAM. For the applications such as Mix0, Mix2, and Mix3, where traffic comes in a burst, the policy g-BUF-DN slows down the packets in the network since the only VC at routers associated to power gated Processing Elements (PE) are busy. This results in network latency overhead. Whereas for the low traffic application such as X264 the policy, g-BUF-DN, shows lesser network latency than proposed policies and shows better latency performance than proposed policies. Table 5.4 gives comparative summary of all policies for 50% dark nodes. The negative values in the table show a decrease in network latency.

5.4.3.2 Performance

Figure 5.14 shows a normalized execution time graph of all proposed policies with respect to the SRAM baseline. The increase in packet latency degrades system performance. Notably, we observe that, on average, the execution time increases over baseline SRAM by 2.2%, 2.4%, 2.4%, and 1% in SRAM-FRQSCL, Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD policies, respectively with 50% dark nodes (*Pattern3*). The SRAM VCs used for critical words in Hy-CRIT-WRD policy lift the performance. As can be observed, all our proposed policies maintain performance with a small execution time overhead.

5.4.3.3 Energy Savings

Figure 5.15 show the total energy and static energy graph for SRAM and proposed policies. The proposed policy SRAM-FRQSCL shows savings in total energy due to the routers running at a lesser frequency. Whereas, we get a significant reduction in static energy of Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD by reason

Pattern	Reference Policy	gBUF-DN	SRAM-FRQSCL	Hy-SEL-ON	Hy-NEIG-SEL-ON	Hy-CRIT-WRD	Row
Pattern2	SRAM	20	2.8	35.4	37.8	37.35	1
	gBUF-DN	-	-21.4	19.3	22.4	21.8	2
	SRAM-FRQSCL	-	-	33.6	36	35.6	3
	Hy-SEL-ON	-	-	-	3.8	3	4
	Hy-NEIG-SEL-ON	-	-	-	-	-0.7	5
Pattern4	SRAM	34.5	3.8	42.84	45.98	44.53	6
	gBUF-DN	-	-47.04	12.67	17.46	15.25	7
	SRAM-FRQSCL	-	-	40.61	43.87	42.36	8
	Hy-SEL-ON	-	-	-	5.49	2.95	9
	Hy-NEIG-SEL-ON	-	-	-	-	-2.69	10

TABLE 5.5: Total energy saving (in percentage) for all policies in 4×4 mesh network.

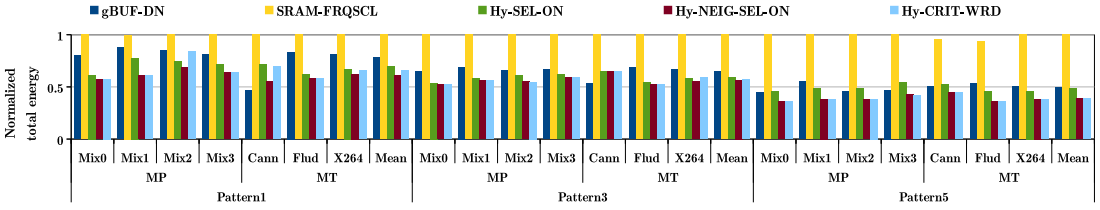


FIGURE 5.15: Normalized total energy of proposed policies with respect to SRAM baseline in 4×4 mesh network (lower is better).

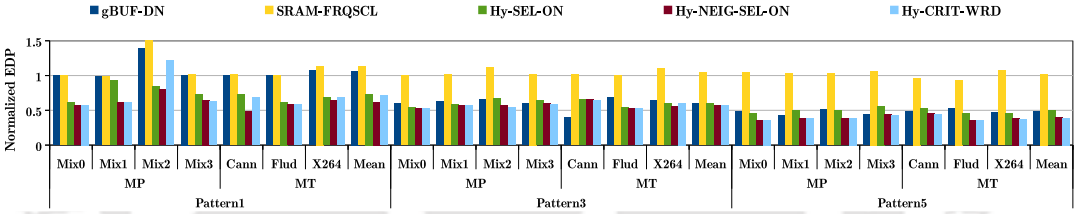


FIGURE 5.16: Normalized EDP gains of proposed policies over SRAM baseline in 4×4 mesh network (lower is better).

of very less static energy of STT-RAM. STT-RAM has costly write operation (cf. Table 5.1), and hence the dynamic energy consumed by STT is more than SRAM. However, lesser static energy gives overall energy savings in STT-RAM-based policies. The energy savings in proposed policies, SRAM-FRQSCL, Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD, are 1%, 52.4%, 58%, and 62.1% respectively, over baseline SRAM with 75% dark nodes.

As can be seen from the graph, for all the patterns and applications, the policy SRAM-FRQSCL shows maximum total energy consumption. The other proposed policies use SRAM and STT-RAM buffers at the router. The policy Hy-NEIG-SEL-ON saves maximum energy since the number of power gated SRAM-based VCs are maximum in it. The policy Hy-CRIT-WRD activates power-gated SRAM

VC when a critical word arrives. These VCs are kept on for a certain number of cycles, and if no critical word comes, it goes in power gate state. This consumes more static energy than Hy-NEIG-SEL-ON. However, based on the application, the total energy consumption differs in benchmarks for Hy-CRIT-WRD.

Table 5.6 gives a comparative summary on power analysis of all proposed policies. The power performance of all the proposed policies are similar to energy consumption given in Figure 5.15. The proposed policies, SRAM-FRQSCL, Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD, shows 0.34%, 43.7%, 45.2%, and 44.7% of power consumption, respectively. Also, the comparative summary of energy and power all policies for Pattern2 and Pattern4 are given in Table 5.5 and Table 5.6. The negative values in the table represent overhead.

Pattern	Reference Policy	gBUF-DN	SRAM-FRQSCL	Hy-SEL-ON	Hy-NEIG-SEL-ON	Hy-CRIT-WRD	Row
Pattern2	SRAM	16.5	0.4	35.5	37.4	37	1
	gBUF-DN	-	-19	22.7	25	24.5	2
	SRAM-FRQSCL	-	-	35	37	36.7	3
	Hy-SEL-ON	-	-	-	2.98	2.4	4
	Hy-NEIG-SEL-ON	-	-	-	-	-0.6	5
Pattern4	SRAM	32.5	0.34	43.7	45.2	44.7	6
	gBUF-DN	-	-47.6	16.7	18.8	18.1	7
	SRAM-FRQSCL	-	-	43.5	45	44.5	8
	Hy-SEL-ON	-	-	-	2.6	1.7	9
	Hy-NEIG-SEL-ON	-	-	-	-	-0.87	10

TABLE 5.6: Power saving (in percentage) for all policies in 4×4 mesh network.

5.4.3.4 EDP Gain

Pattern	Reference Policy	gBUF-DN	SRAM-FRQSCL	Hy-SEL-ON	Hy-NEIG-SEL-ON	Hy-CRIT-WRD	Row
Pattern2	SRAM	2.4	4.8	35.3	38.3	37.8	1
	gBUF-DN	-	2.46	33.7	36.8	36.3	2
	SRAM-FRQSCL	-	-	32	35.2	34.6	3
	Hy-SEL-ON	-	-	-	4.6	3.8	4
	Hy-NEIG-SEL-ON	-	-	-	-	-0.86	5
Pattern4	SRAM	38.18	4.46	41.99	46.52	44.46	6
	gBUF-DN	-	-54.54	6.16	13.5	10.16	7
	SRAM-FRQSCL	-	-	39.28	44.02	41.87	8
	Hy-SEL-ON	-	-	-	7.81	4.26	9
	Hy-NEIG-SEL-ON	-	-	-	-	-3.85	10

TABLE 5.7: EDP gain (in percentage) for all policies in 4×4 mesh network.

The EDP gain is depicted in figure 5.16 for all policies with respect to the SRAM baseline. However, the policies show significant savings in static energy and lead to the major improvement in EDP. On average, proposed policies SRAM-FRQSCL,

Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD shows EDP gain of 1.4%, 52%, 60%, and 62% respectively, over baseline SRAM at 75% dark nodes. Similar to energy analysis, the policy SRAM-FRQSCL shows the least EDP gain across all dark node patterns. This is because of the use of SRAM only VCs that have higher leakage components. From the hybrid buffer-based proposed policies, Hy-NEIG-SEL-ON shows maximum EDP gain since the energy consumption is least due to more STT-RAM VCs being active. Hy-CRIT-WRD also gives gains in EDP on account of better latency (due to the usage of SRAM VCs for critical words) and savings in leakage due to use of STT-RAM VCs for non-critical word packets.

Table 5.7 gives a comparative summary of EDP gain of all policies for Pattern2 and Pattern4. The negative values in the table show EDP loss.

5.4.3.5 Write variation

Reference Policy	Hy-SEL-ON			Hy-NEIG-SEL-ON			Hy-CRIT-WRD			Row
	ctrl	res	data	ctrl	res	data	ctrl	res	data	
STT-SRAM	97.8	98.2	98.57	97.03	98.3	98.3	98.7	99.7	98.7	1
Hy-SEL-ON	-	-	-	-0.78	0.04	-0.3	0.92	1.4	-0.12	2
Hy-NEIG-SEL-ON	-	-	-	-	-	-	1.7	1.4	0.44	3

TABLE 5.8: Write variation reduction (in percentage) for all policies in 4×4 mesh network for Pattern4 (50% dark nodes). (*ctrl*: control VNet, *res*: response VNet, *data*: data VNet)

The write variation is calculated for each virtual network since the type of packets differs for each. Table 5.8 gives the improvement in write variation by our proposed policies over baseline STT-RAM at 50% dark nodes. The policies Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD show a significant reduction in write variation, and it brings it down by 97.8%, 97.03%, and 98.7% respectively. The policies Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD choose minimal written STT VCs for the interval and hence results in even distribution of packets across VCs in the VNet. Similar improvement is seen in other VNets (c.f. Table 5.8). The uniform distribution of writes in STT VCs ensures a longer lifetime of the VC buffers. Note that the negative values in the table show an increase in write variation.

5.4.3.6 Relative Lifetime

Reference Policy	Hy-SEL-ON			Hy-NEIG-SEL-ON			Hy-CRIT-WRD			Row
	ctrl	res	data	ctrl	res	data	ctrl	res	data	
STT-SRAM	5	6.2	6.8	4.7	5.6	5.7	9	7.5	9.8	1
Hy-SEL-ON	-	-	-	0.94	0.9	0.83	1.7	1.2	1.4	2
Hy-NEIG-SEL-ON	-	-	-	-	-	-	1.8	1.4	1.72	3

TABLE 5.9: Lifetime comparison (in times) for all policies in 4×4 mesh network for Pattern4 (50% dark nodes). (*ctrl*: control VNet, *res*: response VNet, *data*: data VNet)

Table 5.9 gives the relative lifetime of all the policies across all VNets at 50% dark nodes. Similar to write variation, lifetime is compared individually for each VNet. The policy Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD show significant enhancement in the lifetime of VC buffer over STT-Base. These enhancements are 5, 4.7, and 9 times for control, 6.2, 5.6, and 7.5 times for response and 6.8, 5.7, and 9.8 times for data VNet respectively. The policy Hy-CRIT-WRD sends critical words through SRAM VC, which reduces the number of writes in STT VCs. This results in less number of writes in STT buffers and hence better lifetime.

5.4.4 Comparison with existing policies

5.4.4.1 Comparison with BlackOut [2] and Hy-Drowsy [1]

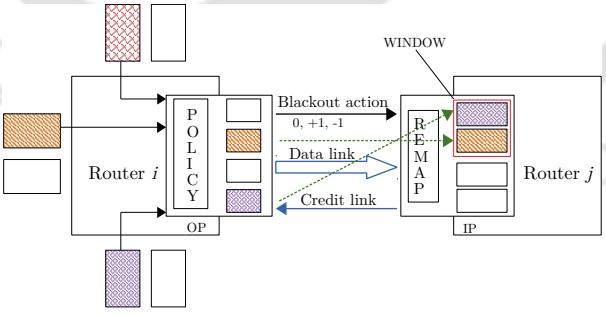


FIGURE 5.17: General overview of the BlackOut [2] router-to-router architecture.

In [2], the authors proposed a power-gating method BlackOut, which individually controls the power supply of NoC buffers. The virtual channel buffers are dynamically switched on and off based on traffic conditions. The BlackOut uses two methods: flow balancing and late binding to control input buffer power gating.

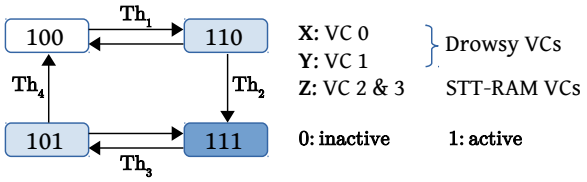


FIGURE 5.18: State transition diagram showing various power-on-off states of VCs in Hy-Drowsy [1] policy.

The flow balancing uses the *output port state* and *active buffer state* to increase or decrease active buffers at a downstream router. Late binding optimizes the allocation of a packet to a buffer using REMAP at an input port of a downstream router. The REMAP binds the physical buffer to different virtual channels.

Figure 5.17 shows router-to-router architecture in BlackOut policy. Here, a dedicated BlackOut action link is used to communicate the REMAP unit at the downstream router for waking up, power gate, or keeping buffers as it is at the port. In the picture, a packet coming to the last VC of the output port of the router i is remapped to the first buffer at the input port of the router j. In the example, only two buffers are active (WINDOW), and other buffers are power gated to save energy.

The Hy-Drowsy [1] uses the combination of SRAM, and STT-RAM-based VC called as Hierarchical buffer architecture. In their proposal, only one SRAM VC is powered-on all the time and based on the traffic load, the drowsy SRAM and power gated STT-RAM are turned ON. The figure 5.18 presents the state transition status of the different levels of VCs with traffic load where the state has encoding (XYZ). Here, the first level (X) represents the SRAM VC which is ON all the time (represented by 1). Whereas, the second level (Y) represents the drowsy SRAM VC which will be ON and OFF based on traffic load. Lastly, the last level (Z) shows the status of power gated STT-based VC. Initially, when the traffic load is low, only the first level SRAM is active (figure, state 100). When the traffic exceeds a certain threshold (say Th_1), the second level (Y) VC gets activated (figure, state 110). On further increase in the traffic (say at Th_2), the third level STT gets activated (figure, state 111). Note that the traffic is measured here based on the buffer occupancy. In case, when the traffic reduces to medium load from the high

load (say at Th_3), the second level (Y) get turned off as the wake-up latency of STT is higher than the drowsy SRAM (figure, state 101). On further reduction in traffic (say at Th_4), the STT-RAM VC gets power gated (figure, state 110).

Policy	Increase in Network Latency (%)	Total Energy saving (%)	EDP Gain (%)
BlackOut	53	12	15
Hy-Drowsy	14	20	27
SRAM-FRQSCL	8	3.8	4.5
Hy-SEL-ON	9.3	42.8	42
Hy-NEIG-SEL-ON	9.7	46	46.5
Hy-CRIT-WRD	7.3	44.53	44.5

TABLE 5.10: Blackout and Hy-drowsy policy details with base comparison in 4×4 mesh network (keeping 50% of the nodes dark).

The comparison of the BlackOut, Hy-Drowsy, and proposed policies with the baseline is shown in table 5.10. The proposed policies increase less packet latency over SRAM design. Whereas, the policies BlackOut and Hy-Drowsy increase latency by 53% and 14% over SRAM. The policy BlackOut shows 12% energy savings and 15% EDP gain over baseline SRAM. Similarly, the policy Hy-Drowsy shows 20% and 27% energy saving and EDP gain. Our proposed policy SRAM-FRQSCL uses only SRAM buffers; hence there is only 3.8% of energy-saving and 4.5% of EDP gain. Our hybrid policies, Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD show 42.8%, 46%, and 44.5% of energy savings and 42%, 46.5% and 44.5% of EDP gain with respect to SRAM.

5.4.4.2 Analysis using Synthetic Workloads and comparison with Router Parking [3]

A selective power gating scheme, router parking [3], presented by Samih et al. proposed three approaches: Aggressive, Conservative, and Adaptive router parking for power gating of the router. In the aggressive scheme, the maximum number of routers is parked or power gated while maintaining connectivity in the network. We compare our proposed policies with Aggressive Router Parking (RP) since it saves maximum power. We performed simulations on a 64-core (8×8) mesh network to study the impact of proposed policies on the packet latency using

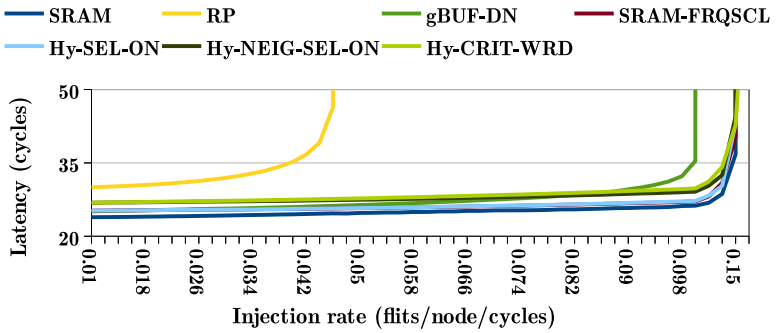


FIGURE 5.19: Performance comparison under uniform random synthetic workloads in 8×8 mesh network (lower is better).

synthetic traffic patterns. We analyze the effect on packet latency by varying the injection rates keeping 50% nodes as dark.

Figure 5.19 shows performance comparison of proposed policies with baseline *SRAM* and *RP* under uniform-random synthetic workload. As expected, the latency of SRAM-FRQSCL policy lies between the latency values for *SRAM* and Hy-SEL-ON, Hy-NEIG-SEL-ON, Hy-CRIT-WRD, and all the proposed policies reach saturation earlier than the *SRAM*. The policy SRAM-FRQSCL shows a very less increase in latency, which is 5.6% on average. The slower running router for the powered off nodes impacts less on latency performance than using NVM at every router to save leakage. The packets take slightly more time to reach the destination as a result of the slow write speed of STT-RAM. These policies make use of STT-RAM VCs, and as a result, the latency grows high by 5.4%, 12.3%, and 12% respectively over *SRAM*. The figure 5.20 shows a performance comparison of proposed policies under bit complement traffic pattern. As expected, the SRAM-FRQSCL, Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD increase latency by 2%, 2%, 8%, and 7.8% respectively. All the proposed policies show improvement over *RP* in latency performance, and the *RP* policy saturates before all proposed policies. This is due to the single path available to source and destination in *RP* policy.

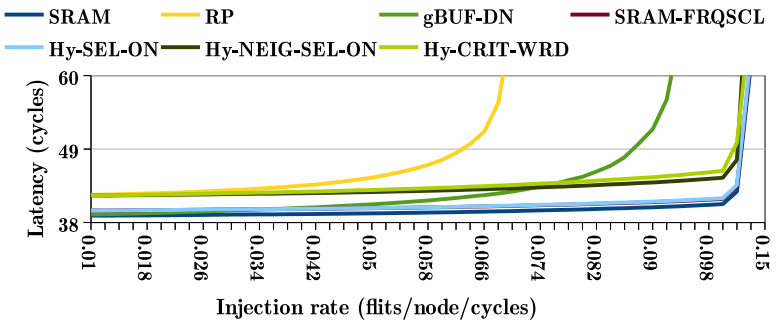


FIGURE 5.20: Performance comparison under bit complement synthetic workloads in 8×8 mesh network (lower is better).

5.4.5 Overhead Analysis

This section briefly demonstrates all types of overheads with respect to the counters and power gating unit in the proposed techniques Hy-SEL-ON, Hy-NEIG-SEL-ON, and Hy-CRIT-WRD. The usage of multiple counters in the proposed techniques incurs the area and storage overhead. The counter values used here can be easily truncated when it reaches its maximum limit. This allows us to use a small counter of size 2 bytes. As per the system parameter given in Table 5.2, a total of 36-byte size SRAM based counters are used at each input port. Hence, for a 4×4 network, this results in counter storage overhead of 4.7% of VC size in the proposed policies.

The area occupied by the STT buffer is one-fourth of the area occupied by SRAM buffers. We use this fact and employ three times the number of SRAM VCs to keep the remaining area for peripheral circuitry and counters of our proposed policies. In particular, the area overhead of counters is 9.4% w.r.t. SRAM baseline design.

The policy gBUF-DN power gates a maximum number of buffers at the router associated with the power-down processing element. To implement this, a power gate controller is introduced at the routers. The Power Gate (PG) controller receives PG signal from the processing unit and informs buffers at the input port to wake-up/power-gate when PE wakes-up/power-gates. The implementation of PG controller incurs area overhead, which is 0.44% as given in [2].

5.5 Discussion

The latency, performance, and energy savings depend on the percentage of dark nodes and the dark nodes distribution on the chip. We have done an analysis of proposed policies for 25%, 50%, and 75% dark nodes on-chip, as shown in figure 5.12. Pattern1 and Pattern2 are an example of a 25% dark node scenario where the dark nodes are contiguous and spatially distributed, respectively. Similarly, the Pattern3 and Pattern4 show an example of 50% dark nodes. Whereas, the Pattern5 depicts a scenario of 75% dark nodes on-chip. The proposed policies show very less latency overhead, while 25% nodes are dark. The latency overhead increases when 50% nodes are dark, and it is maximum when 75% nodes are kept dark on the chip. As it can be observed from the graph (fig. 5.13), the latency overhead is lesser when the dark node pattern is contiguous on the chip. The latency increases for the policies which use more STT-RAM-based VCs since the STT-RAM write takes two cycles, whereas SRAM takes only one cycle (c.f. Table 5.1). Since the policy Hy-CRIT-WRD occasionally uses SRAM VCs for the dark node routers, the latency further improves w.r.t. Hy-NEIG-SEL-ON.

We investigated the replacement of SRAM based router in the context of dark silicon with either frequency scaling them or replacing the buffers with hybrid technology. The main objective is to maintain connectivity in the network and, at the same time, save energy with minimal impact on performance.

Frequency scaling routers attached to dark nodes maintain connectivity and affect the overall latency by around 8%, whereas the use of hybrid technology (SRAM + STT-RAM) affects latency by around 9.3%. In the case of hybrid, if the controller can manage to send critical words earlier using SRAM VCs, the impact on latency is reduced to 7.3%. The existing approaches which use partial resources of the router depending upon the traffic loads result in latency degradation of 14 to 53%. Therefore our proposals of using all the resources at either lesser frequency or manufactured using STT-RAM can save on latency degradation in the context of dark silicon. EDP is the best metric to analyse the energy savings and impact on performance. The use of frequency scaled routers give EDP gains of 4.5% over

baseline SRAM. However, the use of less leakage NVM technology gives the best EDP savings of 42 to 46.5%.

Investigating all our proposals, we can conclude that if we are required to use SRAM based routers, then frequency scaling gives the best EDP instead of powering OFF components and then powering them ON again when traffic load increases. Because powering ON components has an overhead of time and energy. If one can use NVM technology, then the use of hybrid is preferred over pure. The SRAM part of the router helps in maintaining a good latency, and the NVM portion helps in saving leakage. This hybrid technology-based router, while using our proposals, give the maximum EDP gains. Also, as expected, the gains are maximum when 75% nodes are dark. In the case of performance, the impact has to be minimized; we can use the cache optimization of sending critical words first. These critical words are sent via SRAM channels in our proposal, which give the best latency.

5.6 Summary

In the context of dark silicon, several components (cores/memory) are kept powered-off to meet the thermal design constraints of the chip. The NoC is expected to deliver performance maintaining minimal power consumption in the case of dark silicon. The common approaches used for controlling power is to turn-off the routers and wake them up when demanded. However, the wake-up latency adds to performance degradation. This paper investigated different proposals that keep the routers always powered ON, yet help in energy savings. Towards this, we explored the use of frequency scaling and non-volatile memory technologies.

Our first proposal suggested reducing the frequency of routers that are attached to powered down nodes in dark silicon. Reducing frequency saves on dynamic energy and results in a reduction in overall energy consumption. However, reduction in frequency affects the packet latency by around 8% and gives energy savings of 3.8% over baseline SRAM.

For the other proposals, we suggest the use of hybrid memories to build the NoC router buffers. In that, we kept some VCs made using SRAM, and some made using STT-RAM. As STT-RAM is denser than SRAM, we can pack more STT-RAM-based VCs by removing some SRAM based VCs. Only a few among these VCs are used at a time, and the selection depends on the write counts encountered by them. This helps in the distribution of writes evenly across the VCs and helps in handling the weak endurance problem of STT-RAM. The proposal is to use all SRAM VCs and a few STT-RAM VCs when the connected node is powered-ON. In case the node is powered-OFF, we power OFF the SRAM VCs and turn on the same number of STT-RAM VCs. The near-zero leakage of STT-RAM helps in energy savings. The use of SRAM-STT-RAM hybrid helps in controlling latency degradation due to slower writes of STT-RAM. Overall, this policy gives savings of 42.8% in energy with a latency overhead of 9.3%.

We further gave variations to the above proposal by suggesting to use only STT-RAM VCs in the output ports of the neighbouring router connecting to this router. To improve on latency impact if critical words are being sent, then they are sent using the SRAM VCs. Both these policies give energy savings of around 44 - 46% with latency overhead of 7 - 10%.

Our proposals have demonstrated that the use of frequency scaling or non-volatile memory can help in keeping the routers always powered ON even when the associated nodes are powered OFF on account of dark silicon. The proposals help in maintaining connectivity in dark silicon and also save energy.

Chapter 6

Write Reduction Techniques

Whenever writes are incurred by the buffers, our earlier two chapters, distributed them evenly across the buffers, thus improving the overall lifetime of the buffers. Another energy-saving and lifetime enhancing way are to reduce the number of writes happening in the buffers. This can be done by exploring opportunities in data compression and/or sending only relevant data (and avoiding redundant data transmission).

The first proposal is based on compression, where the data is wrapped in a more compact way before it is transmitted or stored. Here we identify the bytes which have zero content and do not transmit them. Necessary metadata is created to enable decoding the data at the receiver. The second proposal is based on dirty data sent by the cache during writebacks. Limiting data-write packets to only dirty flits/words leads to lesser traffic in the network and lesser writes in NVM buffers. This saves dynamic energy and reduces the overall network latency of the packets.

6.1 Introduction

This chapter explores the research direction of write-reduction techniques to reduce the number of writes in NVM buffers. These include compression, where

data are wrapped more compactly before they are transmitted or stored. The data compression techniques reduce overall writes, as well as energy consumption, and improve network latency. However, data compression adds an overhead of encoding and decoding for each data packet irrespective of packet behaviour. To reduce the number of writes in NVM buffers, packet compression is used. In compression, the data are wrapped more compactly before they are transmitted or stored. Intuitively, compression reduces the overall writes as well as energy consumption, capacity and improve the overall latency with the application based performance. Along with these advantages, by employing compression in NVMs designs, the lifetime is also improved. A compression technique requires compressor and decompressor unit either at CPU nodes, in case of cache compression, or at Network Interface (NI), in case of NoC compression.

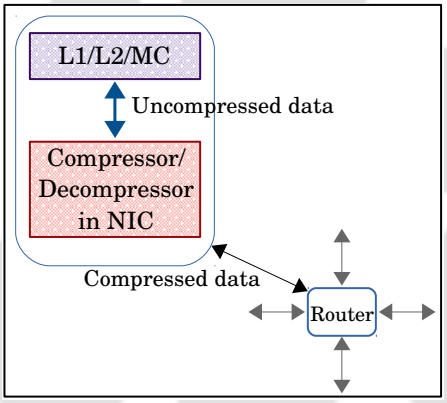


FIGURE 6.1: Network interface controller compression schematic

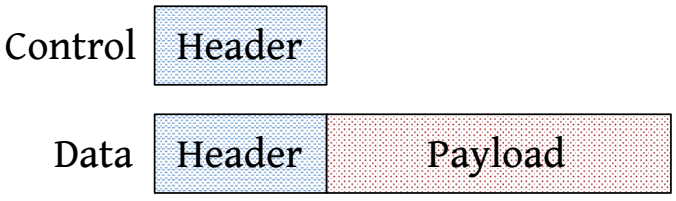


FIGURE 6.2: Flit formats needed for end to end memory accesses through Network-on-Chip

Figure 6.1 shows the network interface controller where the compressor/decompressor fits. Here, at the source NI, the data packet is compressed before flitisization and injected into the network. On the other side, at the receiving NI, the

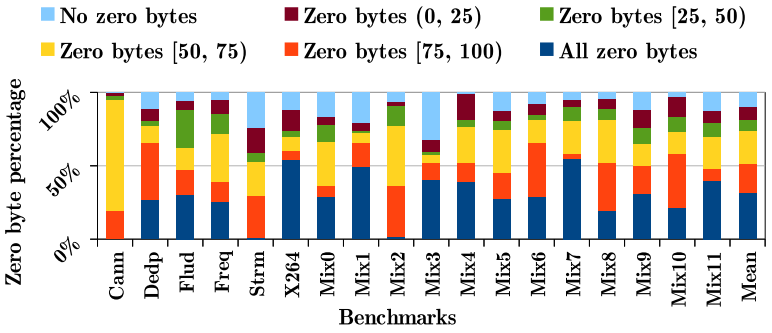


FIGURE 6.3: Zero-Bytes distribution across cache blocks (in percentage) for 4×4 mesh network .

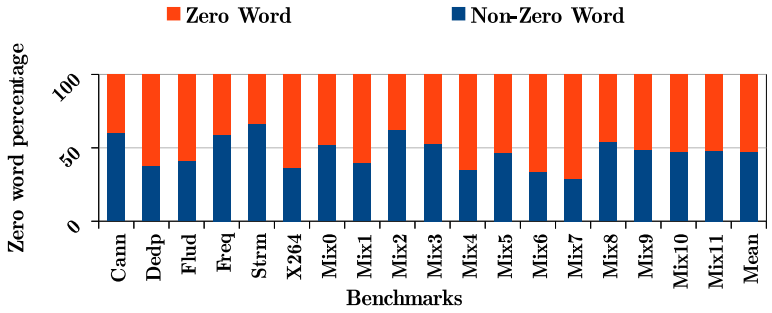


FIGURE 6.4: Zero-Word distribution across cache blocks (in percentage) for 4×4 mesh network (word size 4 bytes) .

flits are decompressed before packetized and sent to protocol buffer. Figure 6.2 shows the two packet format travel through the network for an end to end memory access. The first packet represents the control packet that contains the destination address and a control message. The second packet represents the data packet composed of the header flit (containing the destination and the routing information) along with the body flits (containing the payload information) and the tail flit.

In first proposal, we propose a Network Interface based compression technique that is based on zero bytes present in data packets [179]. To measure the impact of the zero bytes in the data packets, we have conducted experiments, and the results are shown in figure 6.3. The figure shows the percentage of zero-byte(s) present in cache blocks when it is coming into the network. On average, the applications have 29% blocks having all data as zero. On the other hand, the amount of non-contiguous zero-bytes in the cache blocks have the following ranges: blocks with 75-100% bytes as zero are 20%, 50-75% bytes as zero are 23% and 25-50% bytes

as zero are 8%.

Thus, the removal of the large non-continuous zero-data is beneficial for the overall improvement in the efficiency of the network. However, the removal of non-continuous zero-data may result in performance/compression overhead. Another way to compress zero-bytes is to make a small chunk/group of continuous zero-bytes, say zero-word, and remove it. The figure 6.4 shows the zero-word distribution across benchmarks when the word size is 4 bytes. The application *Streamcluster* shows the minimum zero words, whereas the application *Mix7* has the maximum number of zero words. On average, these applications contain 53% of zero-words. In this work, we propose a compression technique that eliminates continuous zero-bytes in a word and non-continuous zero-words in a data packet.

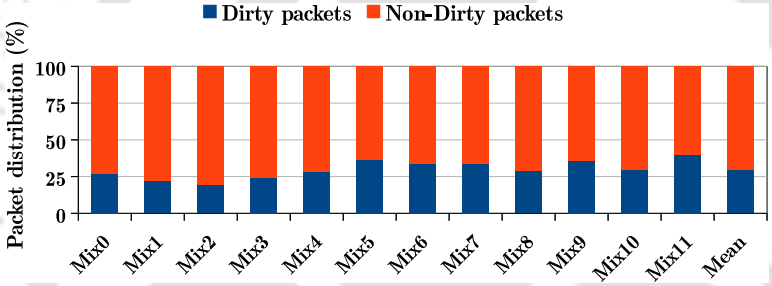


FIGURE 6.5: Total packet distribution in 4×4 mesh network.

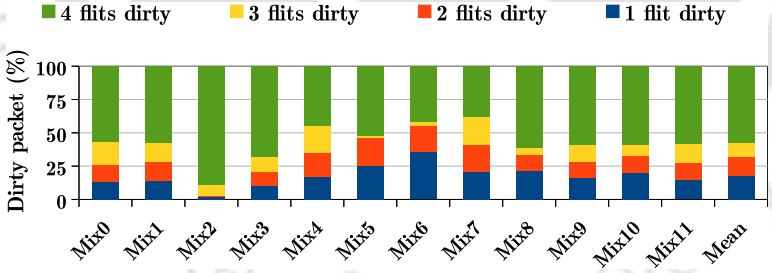


FIGURE 6.6: Dirty flit distribution across cache blocks (in percentage) for 4×4 mesh network .

In the second proposal, we propose a flit based write reduction technique to save the writes in STT-RAM buffers, which is based on dirty flits present in the data packets. We have conducted experiments on SPEC CPU2006 [173] benchmarks to quantify the impact of dirty packets and dirty flits and the results are shown in figure 6.5 and 6.6. Figure 6.5 presents the percentage of dirty packets into the network. Based on our experiments, 70% of cache blocks that travel through the

network are non-dirty. These also include the cache LOAD blocks. The other 30% cache blocks are dirty, and the distribution of dirty flits/words of these cache blocks is shown in figure 6.6. On average, there are 18% of the total dirty packets, which have only one flit dirty. Whereas, packets with only two and only three flits dirty are 14% and 11%, respectively. The remaining 57% packets have all the flits dirty. This leads to the fact that there are many clean flits in the network which are part of the data packets. On average, there are 23% clean flits in the network. Limiting write-back data packets to only dirty flits/words leads to lesser traffic in the network and lesser writes in NVM buffers. This saves dynamic energy and reduces the overall network latency by transmitting smaller size packets.

The major contributions of this chapter are as follows:

- We have proposed the Zero byte-based ENCOding technique, called ZENCO, for the NVM based on-chip interconnect. ZENCO encodes all continuous and non-continuous zero bytes in the data packet by including an isZeroVector with the header of flits belonging to the same packet.
- Further, with ZENCO, we have integrated the Round-Robin based wear levelling technique, ZENCO_RR, to reduce the uneven write variation across virtual channels in a virtual network. The proposed techniques are also applicable to the hybrid-based buffer designs consisting of SRAM and STT-RAM.
- We have also proposed a dirty flit based packet transmission technique, called *DidaSel*, for the hybrid (NVM + SRAM) buffer, based on-chip interconnect.
- At the runtime, depending on the number of dirty flits, *DidaSel* selects whether to send them via SRAM or STT-RAM-based Virtual Channels (VCs).

The chapter is organized as follows: Related works are reported in section 6.2. Proposed write reduction techniques are discussed in section 6.3 and 6.4. Section 6.5 illustrates the experimental methodology and results and analysis. Finally, we conclude this chapter in section 6.6.

6.2 Related Work

In this section, we discuss existing write reduction techniques. Frequent value table-based compression /decompression techniques that dynamically follow the value pattern in the cache or the NoC or in both are proposed in [180]. Boyapati et al. [181] proposed a compression technique that facilitates the approximate matching of data with a controllable value range. Das et al. [182] presented a zero content-based compression/decompression at the network interface. Here, if a packet contains only zero bytes, only a header flit is sent over the network. A coordinated NoC scheduling compression technique that overlaps the decompression latency with NoC queuing delay by integrating data compressors into each NoC routers is reported in [183]. Kim et al. [184] presented a threshold-based, partially applied Huffman encoding technique that reduces the network traffic for extending the lifetime of NoC. A base delta compression technique in the NoC that uses the small intra-variance distance between the flits of the data packet is reported in [185]. Here, the compression module uses the first flit of the data packet as a base to calculate the variance distance for each of the remaining flits. During the flitisization, the base flit is transmitted as it is followed by the variance distance value for the remaining flit. On the other hand, the decompression module reconstructs the data packet by adding the base and the distance value. In *Hy-drowsy* [1], a fine-grained activation/deactivation of different VCs are proposed where each VC is divided into multiple levels. The *Hy-drowsy* uses a combination of SRAM VCs and STT-RAM VCs, termed as the hierarchical buffer architecture. The proposal uses one SRAM VC that is always powered-on and another SRAM VC that can be kept in low-power (drowsy) mode when not in use. Additional STT-RAM-based VCs are kept power gated in low traffic and are turned on during medium to the heavy network traffic load. The SRAM-based VCs are kept in the drowsy state when not in use to save leakage and reduce the wake-up cycles.

6.3 Zero byte based ENCOding: ZENCO

As we have seen in Section 6.1, the share of zero bytes in a cache block is significant. There is a big percentage of cache blocks that either has only zero bytes or more than 50% of bytes as zero (cf. fig. 6.3). Removing these zero bytes from the cache block (data packet) coming to NI will reduce packet size significantly, which results in lesser flits in a network.

The zero bytes present in the data packet can be as continuous byte stream or distributed across the cache block. If an incoming data packet has all bytes zero, only a single flit, i.e., the header is sent to the destination with the information of the data packet being zero. The decompressor uses this information to regenerate the zero packet(s) and forwards it to the receiver buffer. Otherwise, when the packet has a combination of zero and non-zero data bytes, we need some encoding mechanism to compress it.

There are encodings prevalent in the community dealing with sparse data, e.g., sparse CNN [186]. Such encodings are very space-efficient; however, performing them at the runtime would delay the packet transmission as it consumes large cycles for the encoding, nullifying the effect of savings due to compression. In this work, we propose a method of compression/decompression that takes into account zero bytes within the data packet at the granularity of word level. In our technique, when performing the compression, if the zero words are identified in the data packet, then it is not transmitted in the network; otherwise, it is transmitted. Once the compressed packet reaches the destination, the decompressing unit requires information about the positions of the zero and non-zero words. For this purpose, we prepare an n entry binary array (called *isZeroVector*) where each bit or the entry of the array representing whether the corresponding location of the word in the original uncompressed data packet is zero or not. In the compressed data packet, this *isZeroVector* array is added at the front of the packet.

Figure 6.7 shows the sample packet has 4 words and 1 header in an uncompressed form and the packet after compression with the *isZeroVector* binary array. The

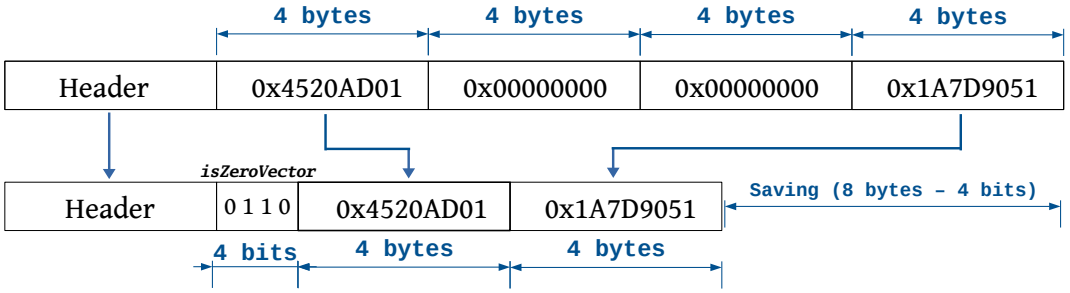


FIGURE 6.7: Packet compression example.

isZero Vector consists of a 1-bit entry for every data word present in the packet. If the first word is zero, then a '1' is written in the first position of *isZero Vector*. If the word is non-zero, then a '0' is written in the corresponding position. As can be seen from the figure, the bit vector has the content `0 1 1 0` as the corresponding words are non-zero, zero, zero, and non-zero, respectively, in the uncompressed data packet. It is also seen that the encoding saves considerable space compared to the original packet. However, the *isZero Vector* array is an overhead, but it is a single bit array and can be easily accommodated in the header as meta-data[1].

Note that the proposed encoding will give a compression ratio depending on the number of zero data words. We can achieve different levels of compression ratio, by performing the encoding at a coarse level (say 8B or 16B at a time) or finer level (say 2B or 1B at a time). The analysis with different sizes of the data words is presented in subsection 6.5.2.11.

6.3.1 Architecture

Encoder: As shown in figure 6.8, for every word we need a zero detection logic. This logic will output a '1' if the word is zero and a '0' otherwise. The output of the logic is stored into the corresponding position of the bit vector. The words that are not zero are copied to the compressed data array. The zero detection logic and the logic to store the output bit in the bit vector operate in a parallel or pipelined way. Hence, the compression needs a total of 2 cycles.

[1]Note that for the transmission protocol which does not demand head and tail flits, the number of flit after compression may increase by one as the *isZero Vector* will be sent with the first flit, and it may change the size of compressed data.

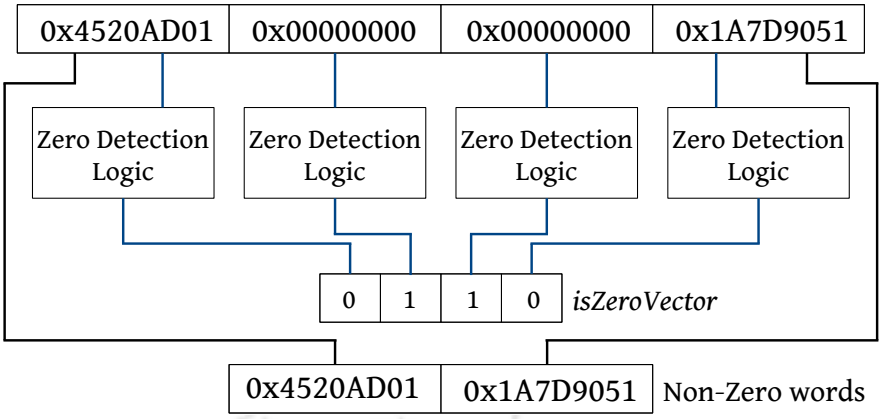


FIGURE 6.8: A single ZENCO compression module

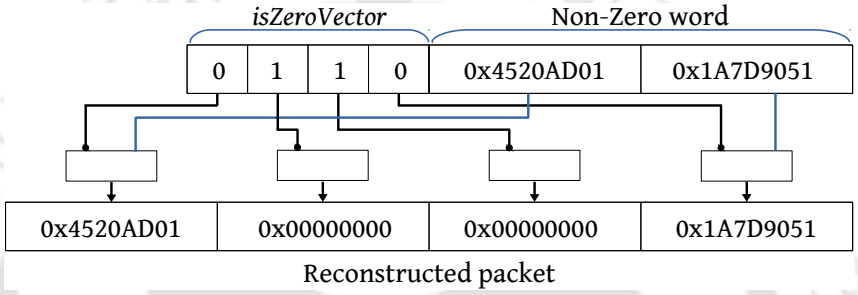


FIGURE 6.9: A single ZENCO decompression module

Decoder: As shown in figure 6.9, for every position in the bit vector, if it is zero, then copy the word from the packet to the decoded final packet. If the bit is one, then skip copying the word in the final packet, and the zero words are inserted. The decoded packet thus constructed and forwarded to the receiver. The decompression takes 2 cycles after applying optimization given in [182].

6.3.2 Write Variation Aware Compression: ZENCO_RR

We propose to use STT-RAM-based buffers in NoC routers. There are two methods to improve write-endurance of STT: i) write reduction technique and ii) wear levelling technique (discussed in Chapter 2) . The compression/decompression falls into the write reduction technique. In the proposed technique, the compression reduces the number of flits by 48% in the network (detailed result and analysis are given in section 6.5). However, this does not improve the lifetime of a buffer since the process adopted by a general-purpose NoC router to allocate VCs at the ports for the incoming flits uses the first available VC. This results in uneven

writes across VCs in a virtual network. In this scenario, the low endurance STT-RAM memory elements lifetime get affected if there are variations in the number of writes.

To mitigate the unwanted write variation in VCs, we integrate the Round Robin based wear-levelling with Zero byte ENCOding, *ZENCO_RR*. We define a write variation and buffer's raw lifetime (Section 2.3.2) as the inverse of the maximum number of writes on that buffer.

Here, to reduce the write variation, we use a round-robin based VC selection method at each input port of a router in the network to allocate the flits into a VC. In particular, at every VC allocation stage, instead of selecting the first available VC, the VC allocator selects a VC among available VCs in round-robin fashion. This selected VC is allocated to the incoming packet in the downstream router. This guarantees that each VC gets a chance to be allocated to some packets after certain cycles.

6.4 Dirty Data based VC Selection: DidaSel

Aim of this chapter is to limit the amount of data that is sent via STT-RAM VCs. This is achieved by identification and transmission of only the modified content within a cache block during writebacks. A writeback involves sending the complete cache block to the next level memory by dividing the block into flits for wormhole routing. As seen from the Section 6.1, figure 6.5, there is a significant percentage of data blocks that have few flits dirty. In particular, when a higher level cache writes back data to lower-level memory, we identify the flits that have changed and only transmit such dirty flits through the network. The dirty only data is provided to the NI by the cache controller. For this, the cache controller maintains extra bits with each block to keep track of dirty data at the flit-level granularity.

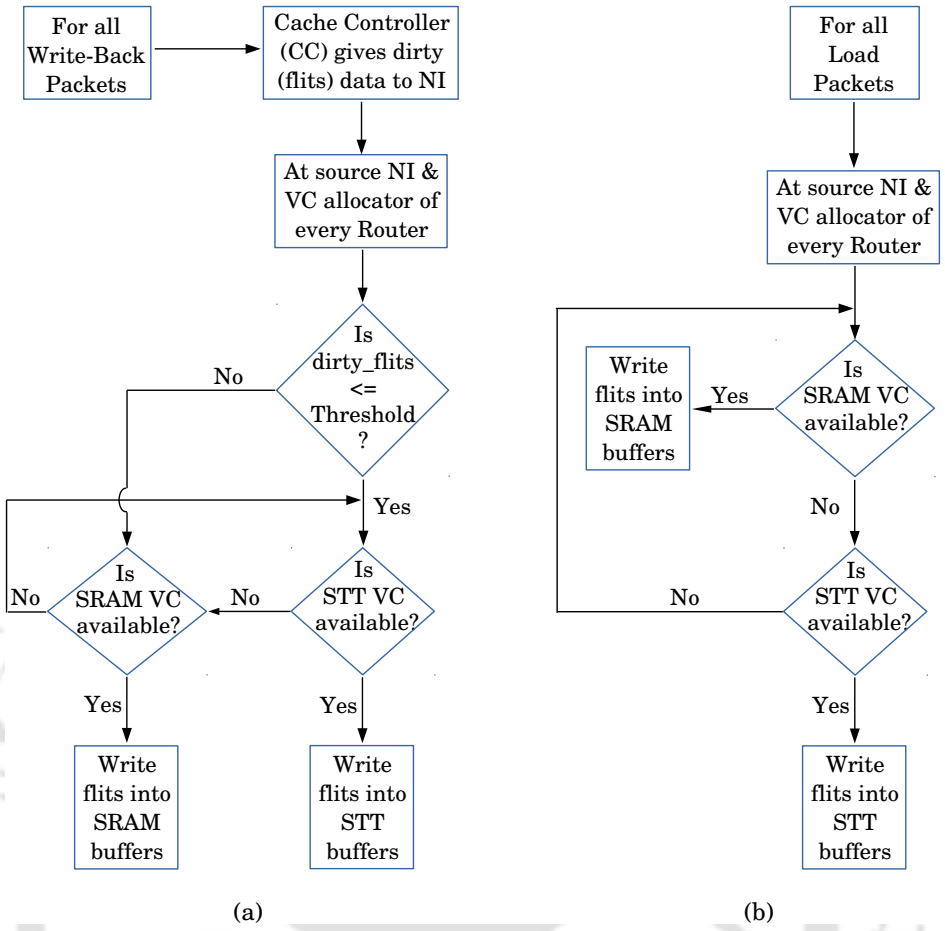


FIGURE 6.10: Flowchart for proposed technique: *DidaSel*. (a) VC allocation strategy for LOAD packets. (b) VC allocation strategy for Write-Back packets

Data traffic passing through the network comprises of loads and stores. For data load(reads), the complete packet must be sent over the network. This is done preferably through SRAM based VCs. If such a VC is not free, then they are sent via STT-RAM VC. For stores (writebacks), the cache controller sends only the modified/dirty flits to NI. The header will have information on which flits are being sent. If the number of dirty flits is lesser than a threshold, then we send them via STT-RAM VCs, otherwise via SRAM VCs. The hypothesis is that, as lesser flits are travelling in place of a complete data packet, even if they travel via a slower STT-RAM pathway, they will not result in network performance degradation. In case the chosen type of VC is not free, then the flits are sent via the first available VC irrespective of its type. Figure 6.10 gives the flowchart of the proposal. The decision of VC allocation is taken at the network interface of the router.

Figure 6.11 shows a working example of the proposal. For this illustration, we

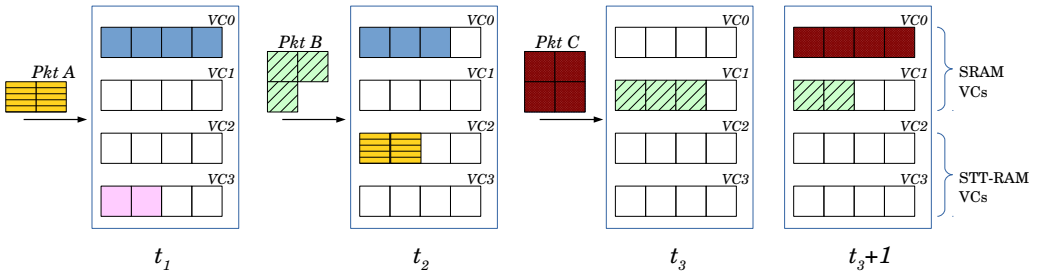


FIGURE 6.11: Working example of proposed technique: *DidaSel*.

have taken the threshold as 2, i.e., if the packet has more than two flits dirty, then they are sent via SRAM else via STT-RAM. The packet pkt_A coming at time t_1 has two flits dirty, and hence it gets written to STT-RAM VC. Whereas, the packet pkt_B coming at time t_2 has three flits dirty hence the packet gets written into SRAM VC. The packet pkt_C is a load packet which will get assigned to SRAM VC. Note that for packet pkt_A and pkt_B are not load packets, and only dirty flits will be sent to the destination.

6.5 Experimental Evaluation

Parameter	SRAM	Drowsy-SRAM	STT-RAM
write latency (cycle)	1	1	2
read latency (cycle)	1	1	1
read energy/bit (pJ)	0.026	0.026	0.022
write energy/bit (pJ)	0.0363	0.0363	0.182
leakage power (mW)	0.206	0.052	0.067

TABLE 6.1: SRAM and STT-RAM buffer configuration

Parameter	Details
core count	16, 64, 2GHz
topology	4 × 4 mesh, 8 × 8 mesh
L1 I & D cache , L2 cache	64KB, 64KB, 4MB (64B block size)
router pipeline	3 stage, 16B flit
packet size	control 1 flit, data 5 flits
virtual network count	3 per port with 4 VC per VNet (ZENCO: 4-STT) (DidaSel: 2-SRAM, 2-STT)
virtual channel depth	1 buffer for control, 4 buffers for data VC

TABLE 6.2: System and Interconnect configuration

Parsec Benchmarks
Canneal (Cann), Dedup (Dedp), Fluidanimate (Flud), Freqmine (Freq), Streamcluster (Strm), X264 (X264)
SPEC Benchmarks
astar (as), bzip2 (bz), cactusADM (ct), calculix (cl), dealII (dl), gamess (ga), gcc (gc), GemsFDTD (ge), gobmk (go), gromacs (gr), h264ref (hr), hmmer (hm), lbm (lm), leslie3d (ls), libquantum (lb), mcf (mf), milc (ml), namd (na), omnetpp (om), perlbench (pb), sjeng (sj), soplex (sp), tonto (to), zeusmp (ze)

TABLE 6.3: Benchmark acronym

16 cores			
Mix	**Workloads**	**Mix**	**Workloads**
0	4×hr, 4×lb, 4×ls, 4×mf	1	4×ct, 4×ga, 4×gr, 4×ze
2	4×ga, 4×gr 4×hr, 4×mf	3	4×dl, 4×ct, 4×ls, 4×ze
4	4×gc, 4×lb, 4×om, 4×pb	5	4×bz, 4×hm, 4×ml, 4×sp
6	4×gc, 4×hr, 4×ml,4×pb	7	4×bz, 4×lb, 4×om, 4×sp
8	4×dl, 4×hr, 4×sj, 4×to	9	4×as, 4×cl, 4×go, 4×na
10	4×go, 4×na, 4×sj, 4×to	11	4×as, 4×cl, 4×dl, 4×hm
64 cores			
Mix	**Workloads**	**Mix**	**Workloads**
0	8×lm, 8×ct, 8×mf, 8×ze	1	8×lm, 8×ct, 8×ls, 8×gr
2	8×ls, 8×gr, 8×mf, 8×ze	3	8×pb, 8×bz, 8×gc, 8×sp
4	8×om, 8×hm, 8×ml, 8×lb	5	8×pb, 8×bz, 8×om, 8×hm
6	8×ml, 8×lb, 8×pb, 8×bz	7	8×dl, 8×na, 8×as, 8×sj
8	8×cl, 8×to, 8×hr, 8×ge	9	8×dl, 8×na, 8×cl, 8×to

TABLE 6.4: Multi-programmed workloads

6.5.1 Experimental Setup

We evaluate our proposed approaches: *ZENCO*, *ZENCO_RR*, and *DidaSel* on a full system Gem5 [168] with Garnet2.0 [169] as the interconnection network model for NoC performance. We use DSENT [170] for router power analysis at 32*nm* technology. The detailed system configuration is given in table 6.2. We use CACTI-STT [171] and NVSIM [172] to get SRAM and STT-RAM latency, read-write energy, and leakage power. Table 6.1 shows all the obtained timing and energy parameters. We evaluate our work with PARSEC [174] and SPEC CPU2006 [173] benchmark suites. The acronym used for benchmarks is given in table 6.3. From the list of SPEC benchmarks, we made twelve multi-programmed workloads for 16 cores and ten multi-programmed workloads for 64 cores. The list of the mix of multi-programmed workload is given in table 6.4. We warm up each multi-programmed workload for 1 billion instructions and run for 100 million instructions.

To show the effectiveness of the proposed techniques, we have compared it with existing techniques [182, 185, 1]. The zero bytes based proposals, ZENCO and

ZENCO_RR, are compared with two methods, *No*Δ [185] compression and the zero content-based compression (*Zero-STT*) [182]. The dirty data based proposal is compared with the three existing methods, the zero flit based compression, *Zero* [182], *No*Δ [185] compression and the hybrid hierarchical buffer based policy, *Hy-drowsy* [1] (see section 6.2). For a fair comparison, we have kept hybrid buffers with *No*Δ and *Zero* policies since the proposed policy uses hybrid buffer. For our proposal *DidaSel*, the threshold[2] is taken as 2 i.e., if a packet has more than two dirty flits, send via SRAM.

Zero policy uses a zero content-based compression at the network interface. Here, a header flit is sent over the network for zero data. Whereas in *No*Δ, the compression is based on the intra-variance distance between the flits of the data packet. The *Hy-drowsy* uses traffic based packet assignment into drowsy-SRAM and STT-RAM VCs rather than using any compression technique. This technique keeps STT-RAM VCs power gated when traffic is low. The SRAM VCs are kept in drowsy mode when not in use to save wake-up cycles of SRAM VCs.

We have also compared with the baseline network having pure SRAM buffers (*SRAM*), pure STT-RAM buffers (*BASE-STT*/*STT*), and hybrid buffers (2-SRAM and 2-STT VCs) (*Hy*). In all hybrid policies, we have used STT-RAM-based first two VCs and SRAM based other two VCs.

As the metrics used for evaluation of ZENCO and DidaSel are different, we present their results in two separate sections. In particular, ZENCO is evaluated based on compressibility of packets based on zero value content, whereas, DidaSel uses the number of dirty flits written back by the higher-level cache(s).

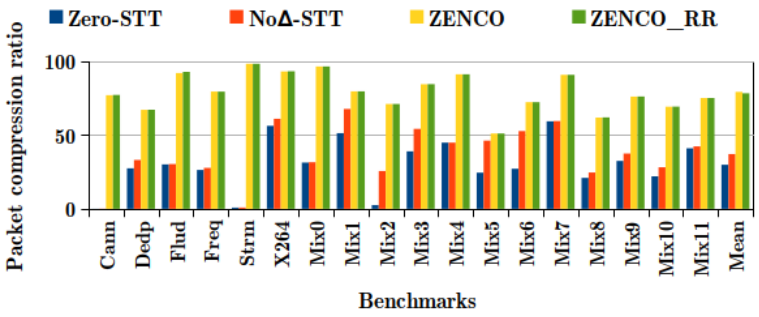


FIGURE 6.12: Ratio of compressible packets in 4×4 mesh network (higher is better).

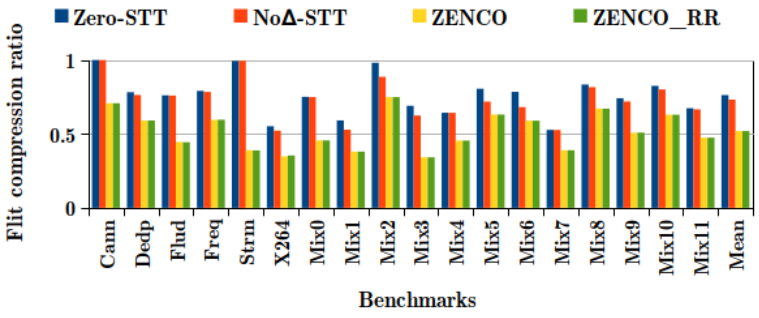


FIGURE 6.13: Flit compression ratio in 4×4 mesh network (lower is better).

6.5.2 Results and analysis for ZENCO and ZENCO_RR

6.5.2.1 Compression Ratio

We define the flit compression ratio as the ratio of compressed data by uncompressed data. It shows the fraction by which uncompressed data reduces after compression. At the same time, the packet compression ratio shows the fraction/percentage of packets that go through compression. Note that the number of packets does not change under compression. Also, after the compression, the flit size remains intact, and only the number of flits that traverse in the network reduces. Hence, the lesser the flit compression ratio and the more the packet compression ratio indicates better compression performance of the technique.

Figure 6.12 and 6.13 show the normalized packet and flit compression ratio respectively for all the techniques. Our proposed techniques, *ZENCO* and *ZENCO_RR*,

[2]Threshold as two is found using empirical analysis on the benchmarks given in the Table 6.4.

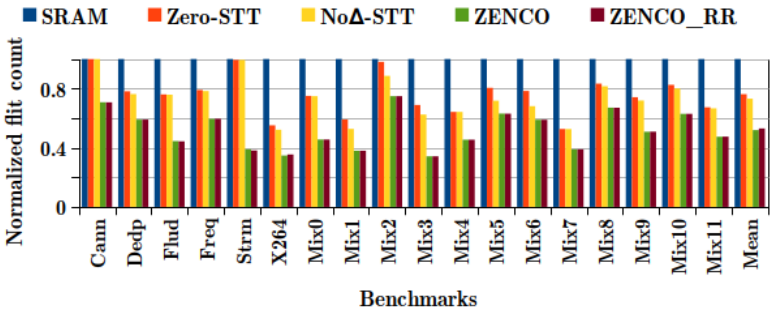


FIGURE 6.14: Normalized flit count for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

show better packet compression of 79% and 78% on average. Whereas, in *Zero-STT* and *No*Δ only 30% and 37% of packets go through compression respectively. The proposed techniques also show better flit compression performance with ratios of 0.52 and 0.53 on average. Whereas, *Zero-STT* and *No*Δ obtain flit compression ratio of 0.76 and 0.73 respectively. The policy *ZENCO_RR* only differs in the write distribution of incoming packets at the router buffers compared to *ZENCO*, and hence, it does not show any impact on compression ratio.

6.5.2.2 Total Flits in network

Figure 6.14 reports the normalized flit count in the network against the baseline. Proposed techniques reduce the number of flits by 48% and 47% over the *SRAM*. Whereas *Zero-STT* and *No*Δ reduce flits by 23.8% and 26.8% respectively. The further improvement in the number of flits over the existing technique is due to the improvement in the compression ratio by the proposed technique over the existing work.

6.5.2.3 Impact on Latency

Figure 6.15 presents the effect of the compression in the network latency with respect to SRAM. The write operation of STT-RAM takes 2 cycles, whereas for SRAM takes only 1 cycle (c.f. table 6.1). Note that according to [56], we have used lower retention time in order to reduce write cycles. A pure STT-RAM based

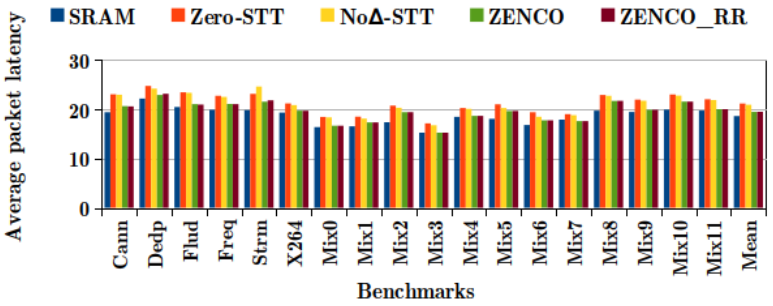


FIGURE 6.15: Normalized packet latency for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

design, *BASE-STT*, shows 19% increase over *SRAM* based design. Our proposed techniques increase latency by 4% (*ZENCO*) and 4.5% (*ZENCO_RR*) with respect to *SRAM* which is an improvement by 12% and 11.8% over the *BASE-STT* design. Whereas, the policies *Zero-STT* and *No*Δ show 4.5% and 5.6% improvement over *BASE-STT* design respectively.

6.5.2.4 Write Variation

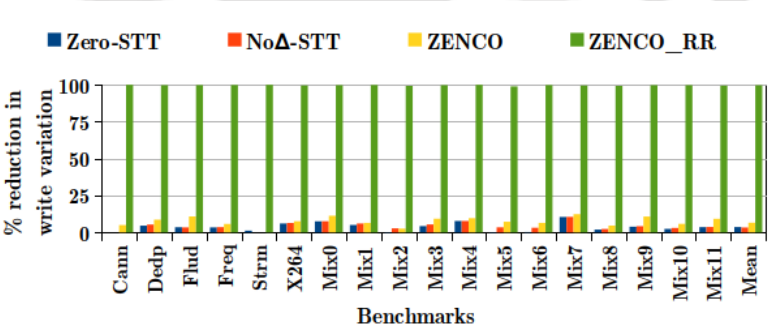


FIGURE 6.16: Percentage reduction in Write variation for all policies w.r.t. baseline *BASE-STT* for 4×4 mesh network (higher is better).

Figure 6.16 shows the reduction in write variation by all techniques with respect to *BASE-STT* design. Our proposed technique: *ZENCO_RR* reduces the write variation by 99.6% over *BASE-STT*. Whereas, the policies *Zero-STT*, *No*Δ, and, *ZENCO* reduces write variation by 3.6%, 3.3% and 7% over *BASE-STT*. The reduction in the write variation is basically due to a round-robin policy for the allocation of VCs by the *ZENCO_RR*, whereas, the existing technique chooses the first available VC for the packet allocation; as a result, the unwanted write

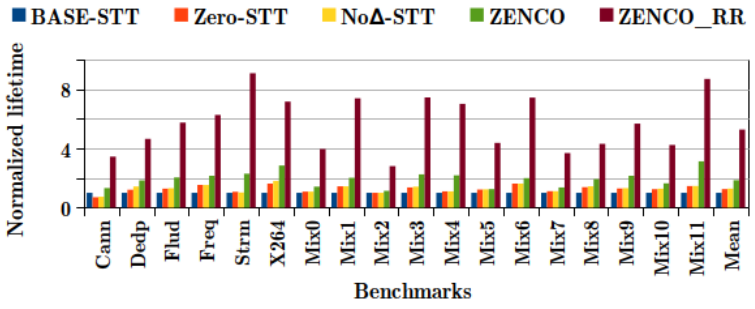


FIGURE 6.17: Normalized lifetime for all policies w.r.t. baseline *BASE-STT* for 4×4 mesh network (higher is better).

variation is generated. However, the lesser number of flits in the network shows a small reduction in write variation without the use of any wear-levelling technique.

6.5.2.5 Analyzing effect on Lifetime

Our policy improves the lifetime by 1.9 times in *ZENCO* and 5.4 times in *ZENCO_RR* over the *BASE-STT*. The policies *Zero-STT* and *No*Δ shows lifetime improvement by 1.24 and 1.27 times over the *BASE-STT* respectively. Compared to *ZENCO*, *ZENCO_RR* further improves the lifetime by 2.9 times.

6.5.2.6 Network Energy Analysis

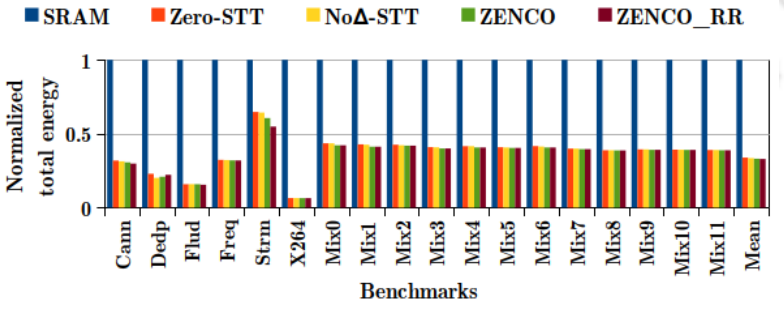


FIGURE 6.18: Normalized total energy for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

The figure 6.18 shows the normalized total energy for all policies with respect to *SRAM* based design. Due to the very less static energy of STT-RAM, we get a

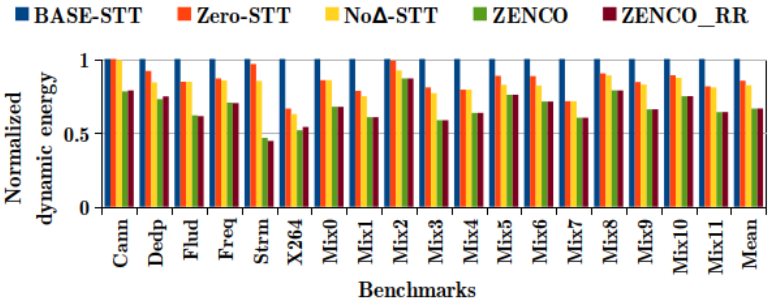


FIGURE 6.19: Normalized dynamic energy for all policies w.r.t. baseline *BASE-STT* for 4×4 mesh network (lower is better).

66% to 67% reduction in total energy in all the STT-RAM-based techniques. The writes in STT-RAM require more energy, which increases the dynamic energy of the system. In figure 6.19, we have presented the comparison of dynamic energy for all compression techniques. In proposed policies, *ZENCO* and *ZENCO_RR*, the dynamic energy is reduced by 18.8% and 18.7% respectively over *Zero-STT* and 15.67% and 15.63% over *No*Δ. This reduction in dynamic energy is due to the lesser number of flits written in the buffer (STT-RAM) compared to existing techniques. Similar dynamic energy reduction can be found in SRAM based buffer design when *ZENCO* and *ZENCO_RR* are implemented. However, in the case of SRAM, the leakage power dominates over the dynamic power, and hence, due to this fact, very little saving is achieved in the total power consumption.

6.5.2.7 Link Utilization

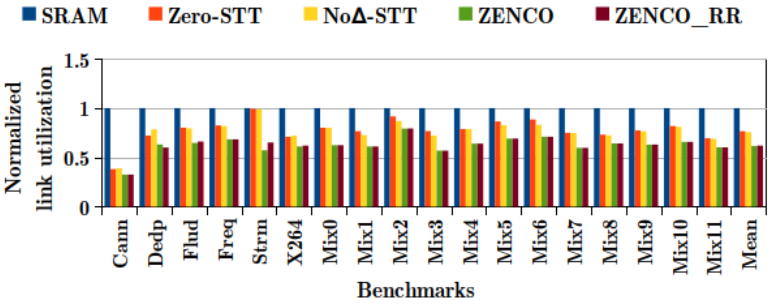


FIGURE 6.20: Link utilization for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

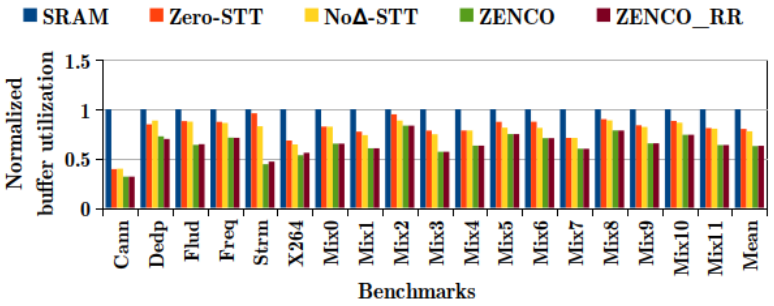


FIGURE 6.21: Normalized buffer utilization for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

The link utilization represents the network load. Lower the link utilization lesser the network become congested. The figure 6.20 shows the average link utilization of the proposed techniques and existing techniques against the *SRAM*. *ZENCO* and *ZENCO_RR* reduce the link utilization by 35.3% and 38% over *SRAM*, 12.6% and 15.38% over *Zero-STT*, and, 11% and 13.7% over *No*Δ.

6.5.2.8 Buffer Utilization

The figure 6.21 shows the buffer utilization reduction of the proposed techniques and existing techniques against the *SRAM*. On average, the *ZENCO* and *ZENCO_RR* reduce the buffer utilization by 37.3% and 37% over *SRAM*, 17.4% and 17% over *Zero-STT*, and, 15% and 14.7% over *No*Δ.

6.5.2.9 Impact on CPI

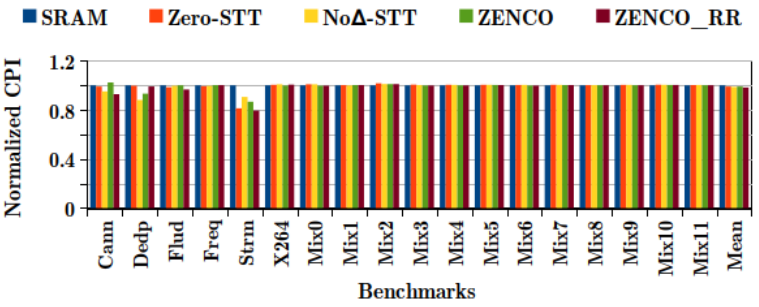


FIGURE 6.22: Normalized CPI for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

The reduction in network latency improves performance. As shown in figure 6.22, the CPI improves by 1% and 1.8% in *ZENCO* and *ZENCO_RR* policies respectively. This improvement is due to the less number of flits travel in the network in the proposed policy.

6.5.2.10 Compression effectiveness on Larger Network

Metric	Zero-STT	NoΔ	ZENCO	ZENCO_RR	Row
Packet compression ratio	50%	56%	90%	90%	1
Flit compression ratio	0.6	0.56	0.37	0.366	2
Reduction in network flit	40%	44%	63%	63.4%	3
% reduction in write var.	1%	1.2%	4.7%	99.3%	4
Normalized lifetime	1.7	1.75	2.2	7	5
Latency increase	9.3%	8.8%	1.5%	0.9%	6

TABLE 6.5: Result analysis for all policies in 8×8 mesh network.

The Table 6.5 shows results for all policies on 8×8 mesh network. Note that the table contains the geometric mean of all simulations run on benchmarks mentioned in Table 6.4. As can be seen, a total of 90% of packets are compressed, giving 0.37 of flit compression ratio in *ZENCO* and *ZENCO_RR*. As expected, the policies *Zero-STT* and *NoΔ* shows lesser compression ratio [Table 6.5, row 1-2, column 3-4]. The number of flits reduced in proposed policies is 63% and 63.4% whereas, policies *Zero-STT* and *NoΔ* reduce flits by 40% and 44% respectively. The policy *ZENCO_RR* reduces write variation by 99.3% and improves lifetime by 7 times.

6.5.2.11 Analyzing effect of Word Size

Network	Word size	Packet compression ratio (%)	Flit compression ratio	Reduction in number of flits (%)
4x4	2	96.5	0.46	54
	4	79	0.52	48
	8	56.5	0.6	40
8x8	2	98.4	0.336	66.4
	4	90	0.37	63
	8	80	0.41	59

TABLE 6.6: Compression word size analysis for proposed policy.

The word size and $isZeroVector$ are important parameters since the compression ratio varies with the size of the word. Also, the size of the $isZeroVector$ depends on the word size chosen for the compression. Note that the packets are compressed

only if there is at least one flit saving due to compression. Smaller word size leads to having more entries in $isZeroVector$, whereas bigger word size reduces the compressibility of a packet since the compressor checks for all bytes as zero in the word. The entries in $isZeroVector$ are in bits, and hence there is not much overhead if the compression word size is small. However, the smaller word size needs to have more hardware-level parallelism to achieve compression as fast as possible. The table 6.6 shows the compression effectiveness for proposed policies for various word sizes. The table shows that for word size 2, there is 96.5% of packets go through compression. Whereas, for word size 4 and 8, the packets compressed are 79% and 56.5%, respectively. The word size 2, 4 and 8 obtain flit compression ratio of 0.46, 0.52 and 0.6 respectively. The less compression ratio here results in lesser packet injection into the network. The compression with word size 2, 4, and 8 reduce the number of flits into the network by 54%, 48%, and 40%, respectively. As can be seen that the proposed policy, *ZENCO*, shows better compression effectiveness in larger (8×8 mesh network) network.

6.5.3 Results and analysis for DidaSel

6.5.3.1 Reduction of Flits in-network:

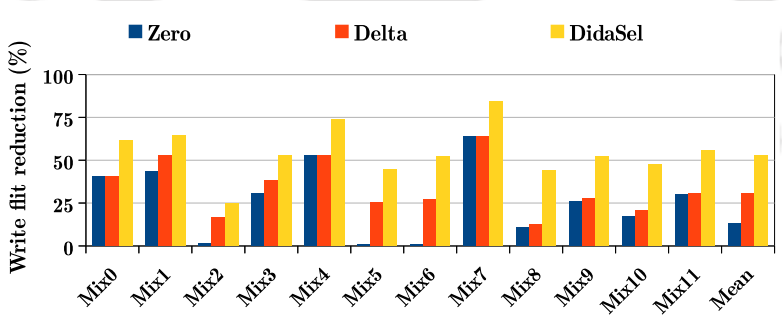


FIGURE 6.23: Normalized write-back flit reduction w.r.t. *SRAM* for 4×4 mesh network (higher is better).

The writeback flits are defined as the flits from writeback cache blocks that go through the network. Whereas, the network flits are all the flits that go through the network (including load and write-back flits).

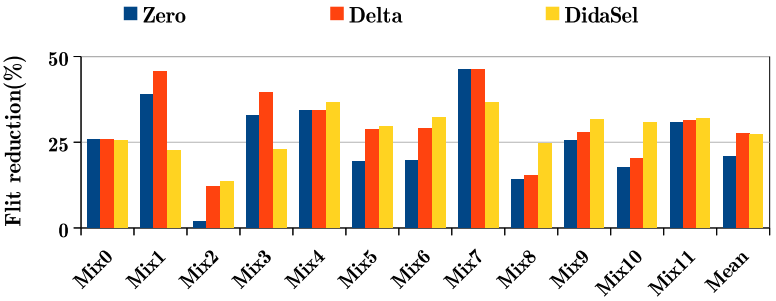


FIGURE 6.24: Normalized network flit reduction w.r.t. *SRAM* for 4×4 mesh network (higher is better).

Figure 6.23 shows the reduction in writeback flits. The proposed policy *DidaSel* shows 52.7% reduction whereas, 13% and 30.7% reduction has been observed by *Zero* and *No*Δ respectively. Figure 6.24 reports the normalized total flit count in the network against the baseline. On average, proposed technique *DidaSel* reduces the total number of flits by 27% over the *SRAM* baseline. Whereas *Zero* and *No*Δ reduce flits by 20% and 27.6% respectively. Note that the policies *Hy* and *Hy-drowsy* has no flit reduction method, hence we have not included the policies in figures 6.23 and 6.24.

6.5.3.2 Impact on Latency:

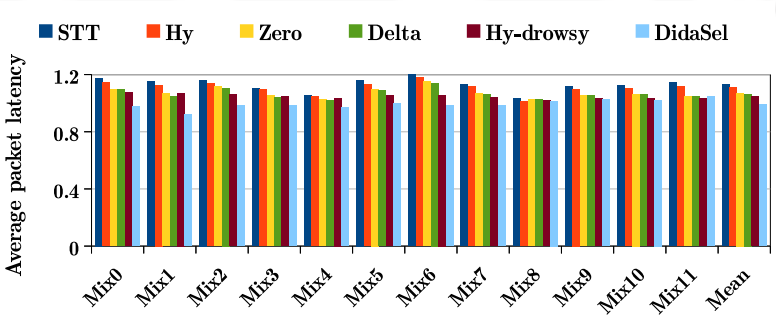


FIGURE 6.25: Normalized packet latency w.r.t. *SRAM* for 4×4 mesh network (lower is better).

The NVM memory technologies, such as STT-RAM, have high write latency. Figure 6.25 presents the effect of the proposed policy on network latency with respect to SRAM. The write operation of STT-RAM takes two cycles, whereas, for SRAM, it takes only one cycle (c.f. table 6.1). Note that according to [56, 57], we have

used lower retention time in order to reduce write cycles. A pure STT-RAM based design, *STT*, shows 13% increase, whereas hybrid design, *Hy*, increases latency by 10% over *SRAM* based design. The proposed technique *DidaSel* maintains latency same as SRAM, which is improvement of 13% over the *STT* design. Whereas, the policies *Zero* and *No*Δ show around 7.2% and 6.5% increase over *SRAM* and, 5.3% and 6% improvement over *STT* design, respectively. In policy *Hy-drowsy*, maximum packets travel using SRAM VC since an SRAM-based VC is always kept active. This improves latency over STT by 8% and causes degradation of 4.6% with respect to *SRAM*. Our proposed technique helps to maintain cycles per instruction (CPI) same as *SRAM*. Whereas *STT* design degrades CPI by 2%.

6.5.3.3 Analysing effect on Lifetime:

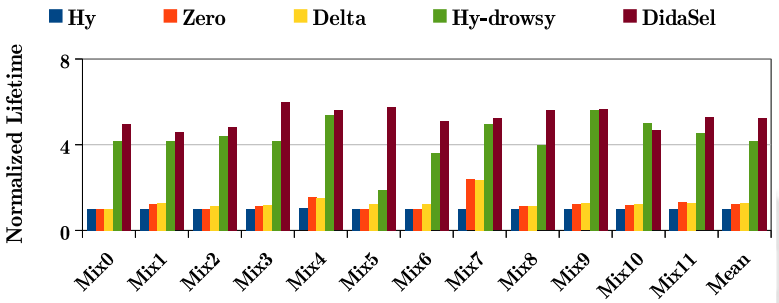


FIGURE 6.26: Normalized lifetime w.r.t. *STT* for 4×4 mesh network (higher is better).

The lifetime of memory can be defined either with a raw lifetime or error-tolerant lifetime [187]. The raw lifetime is determined by the first memory cell that wears out, whereas error-tolerant lifetime is determined by the error recovery methods along with raw lifetime. In this work, we focus on the raw lifetime, which is the base of an error-tolerant lifetime. We have considered a buffer's raw lifetime as the inverse of the maximum number of writes on that buffer.

Our policy *DidaSel* uses a threshold-based flit distribution technique, which limits the number of writes in STT-RAM-based VC. This leads to the improvement of the lifetime of such VCs. On average, *DidaSel* improves the lifetime by 5.25 times over the *STT* baseline (c.f. 6.26). The policies *Hy*, *Zero* and *No*Δ has no VC selection

policy to limit packet allocation to STT-RAM based VCs. However, due to the reduction of the number of flits in the network, the policies *Zero* and *No*Δ show lifetime improvement by 1.4 and 1.5 times over the *STT* baseline respectively. The *Hy* policy does not show any lifetime improvement since the heavily written buffer is the same as in *STT* base design. The policy *Hy-drowsy* activates VCs based on network traffic where an SRAM-based VC is always kept on. This reduces the writes in STT-RAM-based VCs. However, the policy does not support any VC selection method which results in lesser lifetime improvement even though the number of packets assigned to STT-RAM VCs is much lesser in comparison to *DidaSel*

6.5.3.4 Network Energy Analysis:

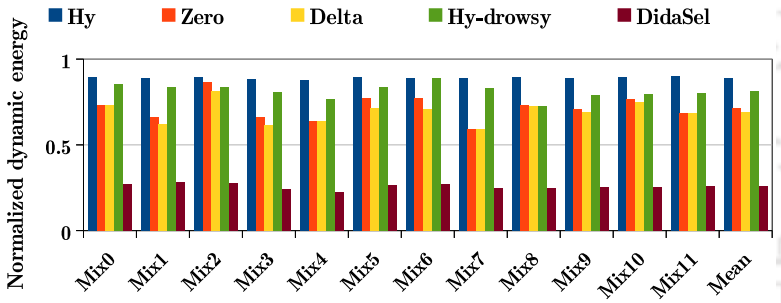


FIGURE 6.27: Normalized dynamic energy w.r.t. *STT* for 4×4 mesh network (lower is better).

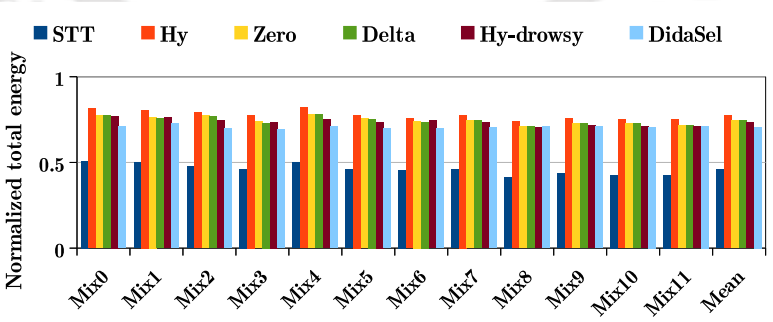


FIGURE 6.28: Normalized total energy w.r.t. *SRAM* for 4×4 mesh network (lower is better).

In figure 6.27, we have presented the comparison of buffer's dynamic energy for all techniques. The writes in STT-RAM require more energy, which increases the

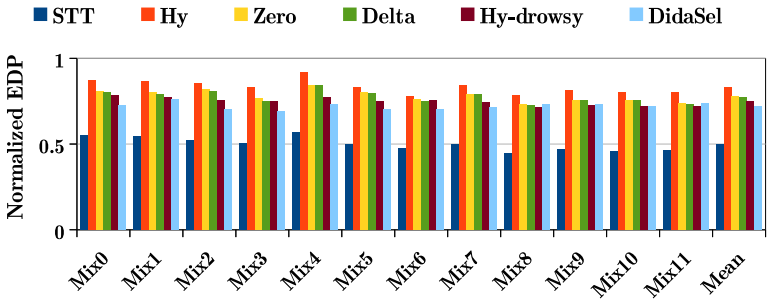


FIGURE 6.29: Normalized EDP gain for all policies w.r.t. baseline *SRAM* for 4×4 mesh network (lower is better).

dynamic energy of the system. The reduction in network traffic leads to dynamic energy savings. In proposed policy, *DidaSel*, the dynamic energy is reduced by 74.28%, in *Zero* by 28.7%, 31.2% in *No*Δ and 20% in *Hy-drowsy* over *STT* baseline.

Figure 6.28 shows the normalized total energy for all policies with respect to *SRAM* based design. Due to very less static energy of STT-RAM, we get 29.4% savings in total energy in *DidaSel* policy and around 25.4% in *Zero* and 25.7% in *No*Δ technique. The policy *Hy-drowsy* uses drowsy SRAM, rather than keeping it power gated, which reduces wake-up latency for SRAM-based VCs. The drowsy SRAM consumes less leakage power during drowsy cycles, which results in total energy savings of 26.5% over *SRAM* based design. Figure 6.29 presents normalized Energy Delay Product (EDP) gains for all policies over *SRAM* baseline. Similar to total energy, *DidaSel* shows EDP gain of 27.88%.

Performance metric	**Flit size (in Bytes)**			**8×8 mesh network**	**Row**
	8	**16**	**32**		
Write flit reduction (%)	58	52.7	29	67.5	**1**
Network flit saving (%)	54	27	22	38	**2**
Latency improvement (%)	10	0.82	-7.4	0.76	**3**
Dynamic energy saving (%)	81.6	74.28	63.5	75.88	**4**
Total energy saving (%)	33.5	29.4	25.8	27.6	**5**
EDP gain (%)	36.88	27.88	20	24	**6**
Column	**1**	**2**	**3**	**4**	

TABLE 6.7: Performance analysis of proposed policy for different flit sizes and effect on larger network of 8×8.

6.5.3.5 Effect of Flit Size

In *DidaSel*, only dirty flits get injected into the network during writeback. The amount of dirty bytes in a flit depends on the flit size. A finer level of extraction ensures less clean bytes in the flit, which results in lesser clean data in the network. Table 6.7 (Column 1-3) shows the effect of flit width on *DidaSel*. The flit size of 8 bytes decreases the number of write-flits in the network by 58%, whereas if we increase flit size to 16 and 32 bytes, the reduction is 52.7% and 29%. The reduction in writeback data results in 54%, 27%, and 22% lesser total flits in the network for flit sizes 8, 16, and 32 bytes, respectively. The write-flits reduction is maximum for smaller flits (8 bytes), which leads to latency improvement of 10% over SRAM baseline. The bigger flits increase clean data in the network and hence latency. For flit size 16 bytes, the latency is almost the same as SRAM design, whereas, for 32 bytes flit size, the latency increases by 7.4% over SRAM design (as can be seen from the row 3, column 3, Table 6.7). The effect on flit reduction impacts the buffer dynamic energy. The more reduction in network flits results in more dynamic energy savings, which is for flit size 8 bytes (Row 4, Column 2, Table 6.7). Similar dependency can be observed for flit size 16 and 32 bytes. Note that the threshold is 2 for flit-size analysis. A similar effect can be observed in total energy savings and EDP gains. Also, note that smaller flit sizes have more overhead at cache block level to keep track of dirty data.

6.5.3.6 Effect on Bigger Network

To show the effect of the proposed policy on a bigger network, we have also performed an analysis of 8×8 mesh network, and the results are given in Column 4 of Table 6.7. On average, the write flit savings is 67.5%, which results in a reduction of total flits in the network. The network flits are reduced by around 38% on average. Also, it has been observed that network latency performance is better than a smaller network. Similar results have been found for dynamic energy, total energy, and EDP gain. Note that the flit size here is taken as 16 bytes.

6.5.3.7 Overhead Analysis

The policy *DidaSel* uses a dirty bit per flit in a cache block to store the clean/dirty status of each flit in the block. These bits constitute storage overhead of 0.72% w.r.t. to baseline system given in Table 6.2. For each write from the processing unit to L1, the corresponding dirty bit is set. When the data is sent from L1 to L2, these dirty bits are examined by the cache controller to extract dirty flits from the cache block before it gets injected into the network by *DidaSel*.

6.6 Summary

The work proposed to use NVM based STT-RAM buffers as they promise high density and low leakage over SRAM counterpart. The major drawbacks of STT-RAM are their costly write operation and low write endurance, which consume more dynamic power and affect the lifetime of the buffers. Towards solving these issues, the chapter proposed three policies to reduce dynamic energy and write variation across the buffers. The first two policies *ZENCO* and *ZENCO_RR* use zero-byte based packet compression, which reduces the number of flits in the network resulting in lesser dynamic power consumption. The policy, *ZENCO_RR*, does the compression with wear-levelling technique to reduce write variation. By experimental evaluation, we found out that *ZENCO* achieve a compression ratio of 0.37 and *ZENCO_RR* enhance lifetime by 7 times with a significant reduction in dynamic energy over baseline. The third policy, *DidaSel*, uses dirty flit based packet traversal, which reduces the number of flits in the network. Depending on the number of dirty data flits, the policy does VC selection. If the number of dirty flits is more than a threshold, then the packet travels via SRAM VC otherwise via STT-RAM VC. This leads to an improvement in network latency and network dynamic power consumption. The full system experimental evaluation shows that *DidaSel* achieves a reduction in writeback flits by 53%, and total network flits reduce by 27%. It shows 5.25 times lifetime improvement of STT-RAM-based buffers

over the *STT* baseline. The policy shows significant improvement in dynamic and total energy consumption.

Thus, by using appropriate compression and wear-levelling techniques, the NVM buffers can become a good choice for the NoC router buffers.

Chapter 7

Conclusion

NoC is the backbone of communication for today's CMPs and also a significant contributor to the power budget. Specifically, the buffers at the ports consume more power in terms of leakage. This thesis proposed to use NVM-based STT-RAM buffers as they promise high density and low leakage. The major drawbacks of STT-RAM are their low write endurance which can affect the lifetime of the buffers and costly write operation. The research work in this thesis is motivated towards improving the utilisation of emerging non-volatile memory technologies and make them a viable option for the NoC router buffers. Towards this, we worked in the following two directions: (i) to extend the lifetime of the NVM buffers affected due to the weak write endurance and unwanted write variations (ii) to overcome the costly write operations and improve the performance and reduce the energy consumption.

Towards solving in the former direction, we proposed two kinds of wear levelling techniques: Intra-VNet wear levelling (methods to minimise the write variation inside the virtual network), and Inter-VNet wear levelling (techniques to reduce the write variation across the virtual network). Whereas, for a solution in the latter direction, we make use of the concept of data compression where a cache block is compressed before it is injected into the on-chip interconnect. In this context, two techniques were proposed in this dissertation that make use of zero bytes and dirty words to compress the data and reduce traffic in the network. This

chapter sums up all the proposed contribution of this dissertation along with the future directions for research.

7.1 Summary of Contributions

- **Write Variation aware wear levelling approach using Static buffer Assignment to Virtual Channels:**

  This contribution satisfies the Objective-1 (c.f. section 1.5) which is as follows: "Minimize write variation across the virtual channels in each VNets."

  Towards this, in this work, we proposed Iso-Capacity and Iso-Area-based VC allocation policies to reduce write variation across buffers. The set of buffers assigned to each VC remain permanently assigned to the particular VC throughout the lifetime of the design. The Iso-Capacity policies WVAR and Hy-WVAR reduce write variation across VCs within the VNets by allocating lightly written VCs to newly arriving packets. This reduces the write variation to almost 0%. The lifetime is improved by 3.9 times and 308 times in WVAR and Hy-WVAR, respectively. The Iso-Area proposal divides the three times resources into sets and presents the scheduling of these sets in the VC allocation procedure. A round-robin allocation of sets (SET-RR) improves packet latency but does not help in reducing the write variation. This reduces the write variation by 55.3% over the baseline policy. The other proposal SET-VNet-WC picks up those VNets from across the sets that have the least, write counts. Write counts are maintained with each VNet, and the selection of VNets to be used for packet transmission is performed at fixed intervals. The policy reduced the write variation by almost 0% and improved lifetime by 11.8 times over the baseline. The hybrid variant of this policy also reduced the write variation and improved lifetime by 1093 times over the baseline.

- **Wear Levelling by Dynamic Buffer Assignment to VCs:**

  The contribution proposes dynamic wear-levelling techniques at the buffer towards achieving Objective-2 (c.f. section 1.5) which is as follows: "Minimise write variation across the VNets in the input port buffers."

  In this proposal, two Intra-VNet wear levelling techniques, Dy-WVAR and Hy-Dy-WVAR, were proposed which aim to further enhance the lifetime of the STT-RAM buffers by distributing the writes in buffers across VNets: Inter-VNet write variation. To control the variation across VNets, we proposed to create buffer groups which were dynamically assigned to the VNets at runtime depending on their write counts. In particular, VNets incurring more write were subsequently assigned to buffer groups incurring lesser writes during an interval. These policies reduce write variation to almost 0% and improve the lifetime by 19.92 and 55.5 times over baseline STT-RAM design. The pure STT-RAM design Dy-WVAR increases latency by 27% over SRAM, which is rectified by the Hy-Dy-WVAR design, which makes the latency 22% over SRAM.

- **Power saving using Frequency Scaling and Power Gating:**

  The contribution proposes design alternatives for buffer management in dark-silicon based CMPs towards achieving Objective-3 (c.f. section 1.5) which is as follows: "In Dark Silicon, when most of the nodes are power-off identify methods to save energy and improve performance while maintaining connectivity."

  This work investigated different proposals that keep the routers always powered ON, yet help in energy savings. Towards this, we explored the use of frequency scaling and non-volatile memory technologies. The first proposal suggested reducing the frequency of routers that are attached to powered down nodes in dark silicon. Reducing frequency saves on dynamic energy and results in a reduction in overall energy consumption. However, reduction in frequency affects the packet latency by around 8% and gives energy savings of 3.8% over baseline SRAM.

  For the other proposals, hybrid memories used to build the NoC router buffers. In that, some VCs are made using SRAM, and some made using STT-RAM. As

STT-RAM is denser than SRAM, more STT-RAM based VCs can be packed by removing some SRAM based VCs. Only a few among these VCs are used at a time, and the selection depends on the write counts encountered by them. This helps in the distribution of writes evenly across the VCs and helps in handling the weak endurance problem of STT-RAM. The proposal is to use all SRAM VCs and a few STT-RAM VCs when the connected node is powered-ON. In case the node is powered-OFF, the SRAM VCs are powered OFF, and the same number of STT-RAM VCs are turned ON. The near-zero leakage of STT-RAM helps in energy savings. The use of SRAM-STT hybrid helps in controlling latency degradation due to slower writes of STT-RAM. Overall, this the policy gives savings of 42.8% in energy with a latency overhead of 9.3%. Further, it is suggested to use only STT-RAM VCs in the output ports of the neighbouring router connecting to this router. To improve on latency impact if critical words are being sent, then they are sent using the SRAM VCs. Both these policies give energy savings of around 45% with latency overhead of 7 - 10%.

- **Methods to reduce write operations:**

  This contribution satisfies the Objective-4 (c.f. section 1.5) which is as follows: "Reduce the number of writes in NVM buffers to reduce costly write operation and improve lifetime."

  Towards solving the issue of the costly write operation, the chapter proposed three policies to reduce dynamic energy and write variation across the buffers. The first two policies ZENCO and ZENCO_RR use zero-byte based packet compression, which reduces the number of flits in the network resulting in lesser dynamic power consumption. The policy, ZENCO_RR, does the compression with wear-levelling technique to reduce write variation. By experimental evaluation, we found out that ZENCO achieves a compression ratio of 0.37 and ZENCO_RR enhance lifetime by 7 times with a significant reduction in dynamic energy over baseline. The third policy, DidaSel, uses dirty flit based packet traversal, which reduces the number of flits in the network. Depending on the number of dirty data flits, the policy does VC selection. If the number of dirty flits is more than a threshold, then the packet travels via SRAM VC otherwise via STT-RAM

VC. This leads to an improvement in network latency and network dynamic power consumption. The full system experimental evaluation shows that DidaSel achieves a reduction in write-back flits by 53%, and total network flits reduce by 27%. It shows 5.25 times lifetime improvement of STT-RAM-based buffers over the STT baseline. The policy shows significant improvement in dynamic and total energy consumption.

Figure 7.1 summarizes the contributions of this thesis.

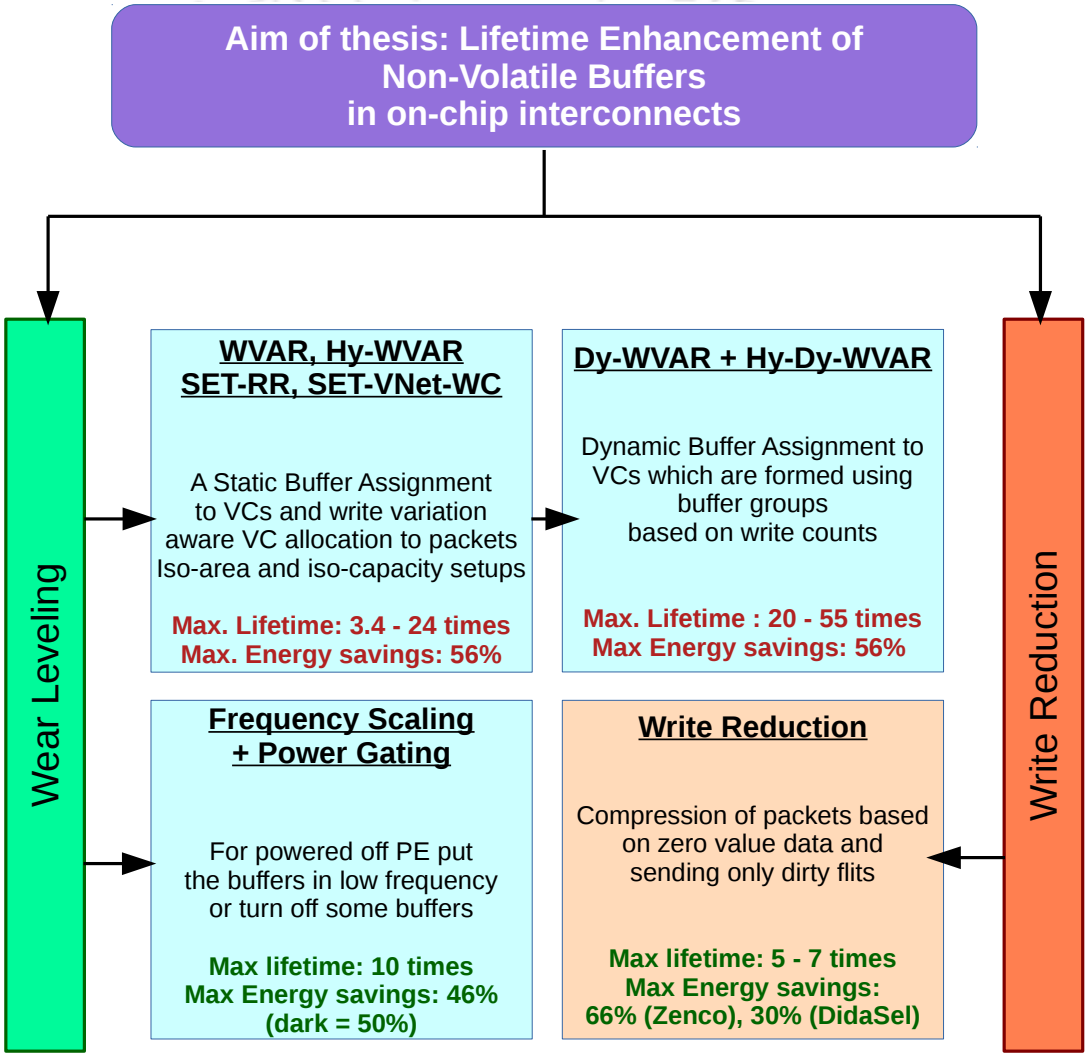


FIGURE 7.1: Summary of the thesis contributions. The results are shown for STT-RAM and hybrid (SRAM + STT-RAM) buffers. Note that, the proposed architectures are also evaluated with various router designs and with different prior works.

7.2 Combination of Techniques

Combination of policies	Baseline policies	Lifetime improvement (times)	Latency improvement (%)	VC type
Dy-WVAR [ch. - 4] + ZENCO [ch. - 6]	Dy-WVAR, ZENCO	1.6, 9.7	15.4, -6	4 STT
Dy-WVAR [ch. - 4] + DidaSel [ch. - 6]	Dy-WVAR, DidaSel	2.24, 2.6	4.2, -8	2 SRAM 2 STT
SRAM-FRQSCL [ch. - 5] + ZENCO [ch. - 6]	SRAM-FRQSCL	-	15.5	4 SRAM
Hy-NEIG-SEL-ON [ch. - 5] + Dy-WVAR [ch. - 4]	Hy-NEIG-SEL-ON	3	-2.9	2 SRAM 6 STT
Hy-CRIT-WRD [ch. - 5] + Dy-WVAR [ch. - 4]	Hy-CRIT-WRD	4.4	-3.5	2 SRAM 6 STT
Hy-NEIG-SEL-ON [ch. - 5] + ZENCO [ch. - 6]	Hy-NEIG-SEL-ON	2	15.8	2 SRAM 6 STT
Hy-CRIT-WRD [ch. - 5] + ZENCO [ch. - 6]	Hy-CRIT-WRD	1.9	13.7	2 SRAM 6 STT
Hy-NEIG-SEL-ON [ch. - 5] + Dy-WVAR [ch. - 4] + ZENCO [ch. - 6]	Hy-NEIG-SEL-ON	3.5	11.7	2 SRAM 6 STT
Hy-CRIT-WRD [ch. - 5] + Dy-WVAR [ch. - 4] + ZENCO [ch. - 6]	Hy-CRIT-WRD	5.13	12.5	2 SRAM 6 STT
DidaSel [ch. - 6] + ZENCO [ch. - 6]	DidaSel	2	4	2 SRAM 2 STT

TABLE 7.1: Relative lifetime and packet latency of combination of techniques with baseline policies given in column II. Note that the negative values in the table shows increase in latency.

In the thesis, we proposed several techniques to enhance the lifetime and reduce the costly write operation of NVM buffers in on-chip routers. Each chapter focused on achieving an objective of either lifetime enhancement or write reduction techniques. In this section, we analyse the combination of techniques and the effect on NVM buffers lifetime and latency performance. Table 7.1 shows the relative lifetime and latency as an effect of combining techniques across chapters with respect to the baseline design. Note that the combination policies were run using the SPEC 2006 benchmarks [12]. The techniques proposed in the thesis, use different system parameters, such as router pipeline stages, VC types, number of VCs per VNet, etc. For the fair comparison, we used baselines parameters with combination policies for benchmark simulations.

The section shows several combinations of our proposed policies on wear-levelling and write-reduction techniques, and their effect on the lifetime and write operation of STT-RAM buffers. The write variation reduction is maximum when Intra-VNet and Inter-VNet write variation techniques are combined. The policy Dy-WVAR uses WVAR (Intra-VNet write variation) across VCs, and Inter-VNet write variation at the input port buffers. Hence, we have shown combinations of Dy-WVAR rather than only WVAR with other techniques. Similar to WVAR, the Iso-Area policies also reduce Intra-VNet write variation. Also, Iso-Area policies were explored only for the large availability of resources; therefore, we have not combined it with other policies.

- **Dy-WVAR + ZENCO**: We observed big improvement of NVM buffer lifetime on the combination of the policies Dy-WVAR (section 4.3) and ZENCO (section 6.3). On comparison with Dy-WVAR policy, the combination policy increases lifetime by 1.6 times, whereas it shows 9.7 times more lifetime with respect to ZENCO policy. The ZENCO policy compresses the cache blocks, which passes through the network which does not get assigned to all VNets. However, the assignment of VNets to buffer groups evens out all the flits over the input buffer. This also evens out the reduced flits in the network, and they are distributed evenly across input port buffers; hence, we get more improvement over ZENCO compression policy. The usage of pure STT-RAM buffers results in increased packet latency and dynamic energy consumption. The compression reduces data packet size and hence the number of flits in the network. This shows a significant reduction in latency over pure STT-RAM buffer based design. The ZENCO compression policy reduces flits in the network; hence, dynamic energy reduction in the combination technique is about the same as ZENCO and 30% with respect to *STT-baseline* policy.

- **Dy-WVAR + DidaSel**: The Dy-WVAR uses WVAR VC allocation and proposes VNet to buffer group assignment to maximise the lifetime of NVM buffers. Whereas, the DidaSel (section 6.4) policy compresses write-back cache blocks and proposes to allocate SRAM or STT-RAM VC to incoming packet based on

the data packet size. This limits the analysis of the combination policy with the hybrid buffer design (used as a baseline). On analysis, we found that the combination policy increases buffer lifetime by 2.24 and 2.6 times over Dy-WVAR and DidaSel designs, respectively. The combination policy shows latency improvement over Dy-WVAR technique due to the reduced packet size. Whereas, it shows latency degradation over DidSel policy due to the VNet to buffer group arrangements at the end of every interval in Dy-WVAR policy. The number of writes in STT-RAM buffers decreases due to decreased packet size. This results in dynamic energy-saving about 72% in comparison to *STT-baseline* policy, respectively.

- **SRAM-FRQSCL + ZENCO**: The SRAM-FRQSCL (section 5.3.2) technique uses SRAM based buffers and reduces or increases the frequency of the router based on associated Procession element power off or on status. This policy does not propose to use pure NVM/Hybrid buffer based design. Hence, we combined only ZENCO techniques for the analysis using SRAM only buffers. The reduction in the network flits observed similar to the ZENCO policy, which results in packet latency improvement of 15.5% over SRAM-FRQSCL policy. Note that we do not show lifetime improvement for this combination technique since it uses SRAM based VCs and SRAM based VCs does not suffer from weak write endurance.

- **(ZENCO + Hy-NEIG-SEL-ON) / (ZENCO + Hy-CRIT-WRD)**: The Hy-NEIG-SEL-ON and Hy-CRIT-WRD methods are used when a few nodes (PE) are dark. Hence the network traffic is always less in comparison to all nodes active and has a running application. However, combining compression technique (ZENCO) further reduces the number of flits in the network, and the number of flit writes in SRAM or STT-RAM VCs. The compression method does not propose any VC allocation, and hence the number of packets assigned to SRAM or STT-RAM VCs remains unchanged. This results in lifetime improvement of 2 times in Hy-NEIG-SEL-ON and 1.9 times in Hy-CRIT-WRD policy. Lesser flit in the network leads to the packet latency saving of about 8−9% in both the policies over SRAM baseline design. The ZENCO compression

policy reduces flits in the network, which results in a dynamic energy saving of $28-31\%$ over *STT-baseline* technique. Note that we compare the combination policies with only hybrid methods Hy-NEIG-SEL-ON and Hy-CRIT-WRD.

- **(Dy-WVAR + Hy-NEIG-SEL-ON) / (Dy-WVAR + Hy-CRIT-WRD)**: The Hy-NEIG-SEL-ON (section 5.3.4) and Hy-CRIT-WRD (section 5.3.5) policies activate either a combination of SRAM and STT-RAM buffers or only STT-RAM buffers based on the router's associated PE status. In addition to this Hy-CRIT-WRD method uses SRAM VC for critical words to improve performance, irrespective of PE status. This results in lesser number of writes in STT-RAM VCs in comparison to Hy-NEIG-SEL-ON method. On combining Dy-WVAR policy, we found that the lifetime of NVM buffers in both the policies, Hy-NEIG-SEL-ON and Hy-CRIT-WRD, shows an improvement of 3 and 4.4 times due to the even distribution of writes across buffer groups. However, this increases the packet latency due to the VNet to buffer group reorganisation overhead of Dy-WVAR technique, which is 2.9 to 3.5 percent in comparison to hybrid methods Hy-NEIG-SEL-ON and Hy-CRIT-WRD. Note that we compare the combination policies with only hybrid methods. The energy consumption in both the combination policies was found similar to the Hy-NEIG-SEL-ON and Hy-CRIT-WRD techniques over SRAM design.

- **(Dy-WVAR + ZENCO + Hy-NEIG-SEL-ON) / (Dy-WVAR + ZENCO + Hy-CRIT-WRD)**: In these combinations, we combined Dy-WVAR and ZENCO compression with both the hybrid policies. The combination policies show maximum improvement over baseline hybrid policy. The hybrid policies VC allocation method uses WVAR at each VNet, and Dy-WVAR does wear-levelling across VNets. In addition to this, ZENCO reduces the number of flits in the network and hence the number of writes in NVM buffers. Also, the combinations show $11-12\%$ of reduction in latency with the combination of hybrid policy in comparison with Hy-NEIG-SEL-ON and Hy-CRIT-WRD design. Similar to other hybrid methods with the combination of ZENCO compression, about 26.3% of dynamic energy reduction is observed over *STT-baseline* design.

- **DidaSel + ZENCO**: Here, we combine both the compression policies. The combination was evaluated for hybrid buffer design since the VC allocation policy proposed in DidaSel technique uses hybrid VCs. The DidaSel policy sends only dirty flits to the next level cache during write-backs. These dirty write-back flits are further compressed at the network interface using ZENCO technique. This reduces the size of the packet significantly. Also, the load data packets, which does not go through compression in DidaSel policy, have a chance of getting reduced in size using ZENCO compression technique. This reduces the total number of flits significantly over DidaSel and ZENCO policies. The compression of packets reduces network flits and writes in STT buffers which results in the lifetime improvement of 2 times in comparison to DidaSel policy. The reduction in the number of flits results in maximum packet latency improvement among all given combination policies in table 7.1. The combination shows around 4% of packet latency decrease over SRAM based design. Due to the significant reduction in flit count, the combination also shows 80% of dynamic energy reduction when compared to *STT-baseline* policy.

7.2.1 Challenges in combining certain policies

The different policies proposed in the thesis uses different VC allocation methods, where the combination of two or more are possible for a few without any modification in the basic proposed architecture. For example, the ZENCO compression technique is given for STT-RAM-based buffers, which can be used with only SRAM or hybrid buffer designs without modification of the compression/decompression logic. Whereas, in DidaSel policy, the design uses hybrid buffers at the input port and the VC selection (STT or SRAM) is made based on the size of the packet. This allows the combination of DidaSel with any other techniques that have only a hybrid buffer design where SRAM and STT buffers are kept active throughout the application execution. On the other hand, the hybrid techniques, Hy-NEIG-SEL-ON and Hy-CRIT-WRD, activates SRAM VCs based on the status of associated PE or critical word arrival. This restricts the combination of Hy-NEIG-SEL-ON

and Hy-CRIT-WRD policy with DidaSel compression technique. However, a few modifications in these policies may make a combination design possible.

The SRAM based policy, SRAM-FRQSCL, can be combined with Dy-WVAR policy as it only distributes writes evenly across VCs and VNets. However, the SRAM buffer does not suffer from weak write endurance. Hence, the combination of these two was not explored.

7.3 Scope for Future Work

The contributions of this thesis can be extended in several ways. Some of these possible future research directions are listed below:

- Our proposed write reduction techniques reduce a significant amount of flits in the on-chip network. However, the compression is done on fixed-size words. In the future, the write reduction can further be explored at variable size words to reduce flits in the network and reduce dynamic energy consumption by STT-RAM buffers.
- Our policies mainly focus on NoC static power reduction by employing STT-RAM as a router buffer. In the future, we can explore thermal awareness sprinting for heterogeneous multi-core NoC in a dark silicon scenario where the routers will get activated based on the temperature to save power consumption, maintaining performance throughput. The idea can also be extended to 3D-NoC.
- Our proposed wear-levelling techniques reduces write-variation at the VC and VNet level. However, the technique does not control/manages the traffic at the router. In the future, buffer allocation with traffic control routing can be introduced for 2D and 3D NoC.
- A conventional on-chip communication employed on CMPs are unable to handle CPU and GPU communication efficiently. In the future, a machine learning-based NoC can be proposed for heterogeneous manycore architectures for efficient communication.

- A power gating method may cause a performance penalty due to unknown traffic patterns. In the future, a machine learning-based power gating method can be proposed to optimise NoC component power gating to maximise power savings.

Appendix A

Simulation Framework

This appendix centres around the exploratory arrangement utilised in our experimental methodologies. The whole investigations given in this thesis are performed on a full-system simulation framework. At a very basic level, the full system framework tests the model of a system as a whole, including CMPs. The framework on which the full-system executes is called *host system* and the virtual environment provided by the simulator for the system model is called as *target system*. Also, the full-system framework structure builds the CMPs by including different modules for the CPU cores, multi-level private and/or shared cache hierarchies, memory controllers and I/O devices. These various small systems are all together connected and communicated by the well known Network-on-Chip (NoC) module that has been attached in the full system framework. The full system simulator enables the system to execute applications independently through an Operation System (OS) installed on the target system. Likewise, the target system permits the kernel modules of the OS to run through the virtual drivers as normally as on a real hardware system.

The simulators are the set of computer programs that run on a host system, and any functionality needed for the target system can be easily modified to meet the design requirement. For example, the conventional NoC buffer made up of SRAM modelled on a full system framework can be changed to NVM buffers or hybrid buffers. For the design space investigation in the target system, the preliminary

parameters like buffer size, VCs per VNet, VC depth, etc. can be easily altered. The brief history of computer architecture simulators is discussed in the next section with their importance in academic and industrial research.

A.1 Computer Architecture Simulators

The computer architecture simulators are a set of software programs that are used to analyse the power and performance of any modelled computer design. The target system can be either (a) full system that simulates the complete computing system, or, (b) target microprocessor called instruction set simulator. There are various simulation techniques among which discrete event simulation, and trace-driven simulation/emulation are commonly used by the researchers [188]. As the emulators are simulators with hardware design constraints, thus, we choose to use discrete event and trace-driven simulators in our experimental analysis.

A.1.1 GEM-5

Gem5 [189] simulator is a combination of the features of GEMS [190] and M5 [191] simulation tools. GEMS has an expanded adaptable memory system that supports cache coherence protocols and a complete interconnection network. M5 gives the assorted CPU models and numerous Instruction Set Architectures (ISA). This subsection briefly discusses the features and the modules of M5 and GEMS along with their limitations.

A.1.1.1 M5

The M5 is an open-source full system execution driven simulator which is an alternative to the commercial Simics simulator. It creates a virtual machine or the complete target system which runs on top of a host framework. M5 simulator is an open-source and goes about as an option in contrast to the business Simics test

system. The intention behind the development of M5 was to analyse the throughput of the interconnect and network protocols by modelling client-server machines. It models various CPU models such as in-order and out-of-order cores with the capability of I/O, memory and OS development. Apart from this, M5 assists by supporting different ISAs, such as ALPHA, ARM, MIPS, Power, SPARC, and X86. M5 is fast enough to execute realistic benchmark suites, like SPECWEB99 [192], Netperf [193], Surge, iSCSI, etc.

The architectural researches are aimed to model next-generation architectures to support variable and changing computing system. In this dissertation, we propose emerging NVM architectures that cope up with the increasing data demands. The memory architecture without any physical overhead can be designed using the full system M5 framework. Although the design and verification of such emerging memories in M5 has constraints such as limited time and space, it is useful enough for the commercial industry. The simulated framework provides the taste of the whole system.

A.1.1.2 Restrictions in M5

M5 has powerful features; however, it lacks having flexible memory system supporting multiple coherence protocols and on-chip interconnect needed to simulate the CMP. In particular, M5 simulates only the point to point snooping based interconnect and caches which are not flexible and scalable for the CMPs. GEMS [190], a timed simulator, is merged on top of M5 to model the complete memory hierarchy with coherence management and on-chip communications using Network-on-Chip (NoC).

Brief details on GEMS are provided in the next section. GEMS cannot work alone without the M5; hence, the functional behaviour is disjoint with the timing models.

A.1.1.3 GEMS

GEMS [190] has two major modules: Ruby and Garnet [194]. Ruby module is responsible for simulating the complete memory hierarchy of CMPs which incorporate L1 cache, L2 cache, memory banks, directories, etc. The components of Ruby is called "machine" and a unique ID: MachineID, is used to identify it. These machines communicate using their MachineIDs through underlying NoC, managed by Garnet. The CMP architecture modelled using Ruby module uses Garnet for on-chip communications. Garnet models the real-time events for transferring packets through the NoC. It also has various network topologies for NoC with variety of design options.

The block fetch request from the M5 processor is passed to the Ruby modules of GEMS. For the fetch request, the very first level of cache,i.e., L1 detects the miss or hit. If the block is found in the first level, the M5 core continues its execution. Otherwise, the block request from the generated core is stalled until GEMS completes its simulation for the miss. The timing dependent functional simulation is controlled and driven by Ruby.

In Ruby, there is a sequencer associated with each L1 which manages the request from the corresponding core. In general, the CMPs have multiple levels of caches that deals the cache requests simultaneously. A controller is attached with each level of cache which manages the functionality of the cache by taking into consideration the consistency and persistence of shared data. The modelling of such controllers is done by SLICC (domain-specific language). The operations and communications of the controller with other machines are managed by SLICC. There are multiple coherence protocols implemented in GEMS. The coherence protocols are governed by the different modules of controllers, and these controllers communicate through message passing which is supported by NoC. The protocol design is the responsibility of SLICC since it is a combined duty of the controllers.

A.1.1.4 CMP Architecture Supported by GEMS

GEMS supports robust SNUCA-based cache architecture which can be configured with different parameters such as cache size, number of banks, hit time, miss penalty, access latency, etc. The parameters are easily modified by making changes in the configuration file. Other than these, additional parameters such as block-size, the number of virtual networks, replacement policy, cache associativity, etc. are set as design demands. In our experimentation, we use the MESI protocol for baseline and the proposed techniques.

A.1.2 GARNET 2.0

With increasing core counts, the on-chip network becomes an integral part of future chip multiprocessor (CMP) systems. The CMPs with hundreds of nodes require a scalable and efficient on-chip communication fabric to work efficiently. To evaluate such system, a detailed and accurate interconnection network model within a full-system evaluation framework is required. To study the combined effect of proposed architecture on the system and interconnection, we use GARNET 2.0 for full-system simulation.

GARNET provides system-level optimisation techniques to be evaluated with a state-of-the-art interconnection network and obtain correct results. It also provides the evaluation of novel network proposals in a full-system design to find out the exact connection of the technique on the entire system. It also enables to implement the techniques which use interconnection along with other components such as cache, memory controller, etc.

GARNET is a detailed interconnection network model inside GEM5. It has a cycle-accurate micro-architecture of an on-chip network router. It provides a packet-switched interconnection where network channels are shared over multiple packet flows. Along with GEMS, GARNET provides a detailed and accurate memory system timing model. It makes use of the Topology and Routing framework provided by GEM5's ruby model.

GARNET models a classic five-stage pipelined router with virtual channel (VC) flow control. It has several micro-architectural detailed components such as flit-level input buffers, routing logic, allocators and the crossbar switch. A router can have any number of input and output ports it needs where very VC at the port has its own private buffer. A credit-based flow-control at every router is supported to meet high bandwidth demands of cache-coherent CMPs. The routing is dimension ordered.

There are various configurable elements of GARNET: i) network topology, ii) interconnect bandwidth, iii) router parameters (such as input/output ports, etc.), iv) routing algorithm, and v) network only simulation. The network topology is configurable in GARNET. The topology of the interconnect is stated by a set of links between the network routers. The GARNET provides point-to-point specific links; hence irregular topologies can be evaluated. GARNET considers the flit size as a phit (physical unit) size. For a different phit size, the on-chip link latencies are modified to incorporate the delay a flit takes to traverse a link. The bandwidth is configured through flit size. Each router in the GARNET has a routing table, and the entries are done at the time of topology formation. The table has entries for output ports and weights corresponding to each destination. To evaluate with synthetic traffic types, GARNET provides several traffic patterns such as uniform random, bit-complement, nearest neighbour, transpose, etc. Since the synthetic traffic patterns are used for an estimate of the network performance, GARNET is designed to run in network-only mode also.

GARNET outputs various statistics: total number of packets/flits injected into the network, link utilisation (average and per-link), average load on a VC, average network latency (inside the network as well as queuing at the interfaces), and number of reads and write at every router. Also, various counters can be added easily to the modules and statistics displayed.

A.1.2.1 Result Analysis

As GEM5 is a full system simulator, it can run a real set of applications on the simulated architecture. During the experiments, GEM5 logs the various statistics of the running application(s). Some of the important information needed in this dissertation are described below:

- Total Cycle Executed: The metric records the summation of all the cycles executed for all cores. Besides this, GEM5 also collects the executed cycles (comprising of busy and idle cycles) for the individual core.
- Total Simulated Instruction: This collects each cores number of executed instructions as well as outputs the summation of total instructions executed.
- L1 Demand Access: GEM5 also records the demand accesses including demand hit and miss for each L1 bank private to a core.
- L2 Demand Access: It records the individual demand hit and misses for each shared L2 bank.
- Average Network Latency: It is the average time spent by packets in the network.
- Average Packet Latency: This value represents the sum of average network latency and average queuing latency.
- Buffer Reads: It is the value recorded per router. It represents the total number of reads that happened in a router buffer.
- Buffer Writes: It is the value recorded per router. It represents the total number of writes that happened in a router buffer.
- Crossbar Activity: It shows the number of switch traversal happened in a router.

Other than that, we have also added some of the additional metrics that are needed to analyse the conducted simulations for the NVM and hybrid buffers:

- VC Write: The metric outputs the value of the counter associated with a virtual channel. It is used to calculate the lifetime improvement (as given in Equations 2.1 and 2.2) by the proposed approaches.
- Buffer Write: This outputs the value of counter associated with a buffer group. It is used to calculate the lifetime improvement (as given in Equations 2.3 and 2.4) by the proposed approaches.

Apart from these metrics, the other metrics like Cycle Per Instruction (CPI), Instruction Per Cycle (IPC), Miss Per Thousand Kilo Instructions (MPKI), etc. are easily derived or calculated from the given documented metrics provided by GEM5.

A.1.3 NoC Power Modelling Tools: DSENT and Orion

With the ever-increasing demand for on-chip network bandwidth, the power efficiency of the interconnection network has become more important for CMPs. We have used two power modelling tools i) Orion2.0 [25] and ii) DSENT to investigate power performance of proposed techniques in our thesis.

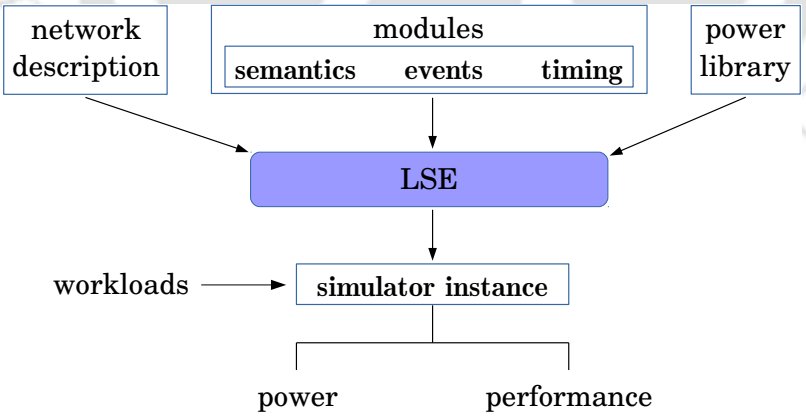


FIGURE A.1: Design, modelling and simulation process under LSE.

Orion: Orion [195] is a network power-performance simulator which can be integrated with application simulator or can run several workloads on the fabric and analyse their impact on overall network power and performance. Orion uses LSE (Liberty Simulation Environment) [196] as a basic simulation infrastructure. The

physical hardware blocks are modelled as logical functional modules which communicate via ports. These ports are used for data transfer between the modules via message passing. Each module has its own parameters and control functions, for example, a buffer module has ports corresponding to read/write ports and parameters such as buffer size and width.

An interconnection network is composed of messages generating, transporting and consuming agents. For example, sources are message generators, routers and links are transporters and sink consume the message. The selection of building blocks (modules) of an interconnection network is guided by a hierarchical modelling methodology [197]. The basic components are message sources and sinks, router buffers, crossbars, arbiters and links. Conceptually the modules are divided into two classes. The modules from the transporting class do not store or modify message; whereas the processing class generates, stores or modifies the message content.

Orion uses CACTI [198] to compute the actual C_g, C_d, and C_w values where C_g is gate capacitance of transistor/gate, C_d is diffusion capacitance of transistor/gate and C_w is capacitance of metal wire. The transistor sizes can be user-input parameters, or automatically determined by Orion with a set of default values from CACTI [198] and applied with scaling factors from Wattch [199]. Driver transistor size, e.g. crossbar input driver, is computed based on their load capacitance. The power models, for crossbars and arbiters, are explained in detail in [39]. The Orion power models are based on the detailed estimation of gate and wire capacitance and switching activities rather than estimating power based on transistor count and area. The models are constructed in a hierarchical fashion to maximise the reuse of power models.

ORION was one of the first NoC power models released and widely used for power estimation of NoCs. However, validation against recent NoC prototypes, Intel 80-core Teraflops chip and the Intel Scalable Communications Core (SCC) chip, there was a significant digression in power-performance that can lead to incorrect NoC design choices. An extensive enhancement of the original ORION models, ORION

2.0 [25], includes completely new subcomponent power models, area models, as well as improved and updated technology models.

DSENT: DSENT (Design Space Exploration of Networks Tool) [170] is a unified framework for photonics and electronics which enables cross-hierarchical area and power evaluation of optoelectronic on-chip interconnects. The framework is for electrical NoC components featuring integrated timing, area, and power models. In particular, it focuses on the impact of network utilisation, technology scaling and thermal tuning.

DSENT rely on a minimal set of technology parameters that encapsulate the major characteristics of deep sub-100 nm technologies without the concern of transistor modelling. DSENT supports 45 nm, 32 nm, 22 nm, 14 nm and 11 nm technology nodes currently. Technology parameters are extracted using SPICE models for the 45 nm node. The models for 32 nm node and below are projected [200] using the virtual source transport of [201] and the parasitic capacitance model of [202]. A switch from planar (bulk/SOI) to tri-gate transistors is made for the 14 nm and 11 nm nodes.

Similar to Orion, DSENT can also be used standalone or can be integrated with an architectural simulator [194, 203]. DSENT can also be used to generate traffic-dependent power-traces and area estimations for the network [204].

DSENT also provides on-chip photonic network modelling, which consists of photonic devices along with electrical interface circuits. The detail can be found in [170].

A.1.4 Timing and Power Modelling Tools: NVSim and CACTI-STT

The GEM5 full system framework simulates a real set of applications by utilising the underlying architecture. However, this framework does not model timing, power, and area for different memory technologies at the cache or memory level.

CACTI 6.0 [205] and NVSim [172] are two well known high-level modelling tools in the computer architecture research community. The tools take some of the architectural parameters like cache memory technology, cache size, cache associativity, cache level, block size, access technique (UCA or NUCA), etc. to simulate the cache at device/circuit level. CACTI 6.0 models traditional as well as non-uniform banked caches and memories using SRAM, and DRAM memory technologies. However, it does not support STT-MRAM technology. CACTI-STT [171] extends CACTI to support in-plane STT-MRAM technology. Additional parameters are combined with existing analytic models and integrated with CACTI by modelling bit-line, read circuitry, delay, area and energy consumption.

NVSim models different emerging NVM memory technologies such as STT-RAM, ReRAM, PCRAM, and the NAND flash. NVM outputs the power consumption, area overheads, cache access time, etc. by modelling the architecture at the device level. Based upon the ITRS reports [206, 207, 34], the NVM and the SRAM memory technology fabricated caches can belong to any three categories based on the power and performance modes: i) HP: known as High Performance cell which consumes large power and very fast in the access operation; ii) LSTP: Low STandby Power cells, incurs low power when idle. However, their accessing is slower than the HP as the transition from the low power standby-mode to active-mode incurs extra cycles; iii) LOP: Low Operating Power cells which incur less power both in standby as well as active mode. It is the slowest among the three methods. Furthermore, CACTI-STT and NVSim support fast, sequential, and normal types of cache access techniques. We have used fast-mode of accessing in our work. The transistor length is an important parameter here, and it is also known as the technology parameter in CACTI-STT and NVSim. We have used a transistor with the channel length of 32nm and 45nm and the temperature of 350K in our work. Note that the version of NVSim [172] and CACTI-STT [171] does not support the impact of operating temperature and the write errors of STT-RAM.

The detailed parameters used for simulation are given in table A.1.

NVSim models dynamic energy and leakage power consumption as follows:

Parameters	**Value used**
Technology	32 nm,45 nm
Access mode	normal
Power/performance mode	HP
Temperature	350K
STT-type	in-plane
MTJ-area	1.8 e^{-10}
Delta	16.12
Buffer size	1KB
Retention period	10ms

TABLE A.1: 1-KB SRAM and STT-RAM buffer configurations.

$$ENERGY_{dynamic} = CV_{DD}^2 \tag{A.1}$$

$$POWER_{leakage} = V_{DD}I_{leak} \tag{A.2}$$

$$Delay = \tau\sqrt{(\ln\frac{1}{2})^2 + \alpha\beta} \tag{A.3}$$

where α is the slope of the input, $\beta = g_m R$ is the normalised input transconductance by the output resistance, and $\tau = RC$ is the RC time constant. NVSim models both gate leakage and sub-threshold leakage currents in I_{leak}.

CACTI-STT models read and write energy per operation by evaluating the Equations A.4 and A.5. Here, C_{tot} depends on the total capacitance of the bitline, on the all wire contributions and on the access transistor. V_{read} and V_{write} are the read and write voltage respectively. R_{MTJ} is the MTJ resistance, R_{ACC} is the NMOS resistance, and τ_{write} is the MTJ switching time.

$$ENERGY_{read} = C_{tot}V_{read}^2 \tag{A.4}$$

$$ENERGY_{write} = \frac{V_{read}^2}{R_{MTJ}R_{ACC}}\tau_{write} + C_{tot}V_{write}^2 \tag{A.5}$$

$$R_t = \tau_0 e^{\Delta} \tag{A.6}$$

	A	**B**	**C**
STT-Type	In-Plane	In-Plane	In-Plane
Jc0 $[mA/cm^2]$ (Critical current at zero temperature)	2	2	2
Δ (Thermal Stability)	40.29	40.29	40.29
MTJ Area $[cm^2]$	2×10^{-10}	2×10^{-10}	2×10^{-10}
Rp $[k\Omega]$ (MTJ resistance in parallel magnetization)	1.5	1.5	1.2
Rap $[k\Omega]$ (MTJ resistance in anti-parallel magnetization)	3	3	1.8
Vbitline $[V]$ (Write voltage)	1.8	1.3	1.8
Raccess $[k\Omega]$ (Equivalent resistance of the access transistor)	1.5	0.3	0.3

TABLE A.2: MTJ configurations.

The data retention time of an STT-MRAM bit-cell depends on thermal stability (Δ) of the MTJ. It is usually evaluated by Equation A.6.

In [171], the authors have generated high-performance and low-power cache memories for three different MTJ configurations and compared all with SRAM technology. The MTJ input parameters used are given in Table A.2 [171]. The high-performance cache was generated for 8 way set-associative memory without any error correction mechanism which ranges in size from 32 kB to 512 kB. It has been observed that The configurations A and B have high write time compared with SRAM. In configuration C, the parallel and anti-parallel resistances are reduced, which results in write latency as SRAM. All three configurations show higher write energy consumption than SRAM.

A.2 Benchmarks

As discussed in the previous sections, the full system simulator GEM5 runs real benchmark applications on a designed architecture. The performance of the architecture is evaluated based on the stats collected from these simulations. The power-performance stats of the simulated architecture gives sufficient information to the researcher and the hardware manufacturer about the real-world behaviour about the design. There are several benchmark suites such as PARSEC [11], SPEC CPU 2006 [12], SPLASH-2 [208], etc. to facilitate simulation on GEM5. In this dissertation, we have used the multi-programmed SPEC CPU 2006 benchmark and multi-threaded PARSEC benchmark suite to test our proposed architecture design. The detailed description of these benchmark suites are given below:

A.2.1 PARSEC

The Princeton Application Repository for Shared-Memory Computers, (PARSEC) [11], benchmark suite is developed for next-generation CMPs' evaluation and validation. PARSEC is a collaborative project with Intel to develop benchmark applications which help the research community for efficient design of future computing systems. The benchmark suite is available as open-source and is widely accepted in the architecture research community. It is used in industrial as well as academic research. Some of the key objectives of PARSEC are given below:

- Next-generation applications for various real-world problems.
- Distinguished input sizes for every application.
- Multi-threaded applications.

Earlier benchmarks were application-specific and executed in the serial manner [11]. PARSEC version 2.1 consists of 12 applications, and each application is multi-threaded and parallelised. The PARSEC applications are selected from diverse real-world areas such as computer vision, animation physics, finance, media processing, etc. The table A.3 gives description of PARSEC benchmarks. In general, multi-threaded application exchanges data between its spawned threads. The table A.4 reports the data sharing and exchange description of these applications.

The term benchmarks are also alternatively called workload, application or program.

In PARSEC, each benchmark has its own working set of input sizes: large, medium, small, test, etc. Users can run workloads with any input set size based on their architecture design and its requirement.

Program/ Benchmarks	Application Domain	Parallelization		Working-Set
		Model	Granularity	
blackscholes	Financial Analysis	data-parallel	coarse	small
bodytrack	Computer Vision	data-parallel	medium	medium
canneal	Engineering	unstructured	fine	unbounded
dedup	Enterprise Storage	pipeline	medium	unbounded
facesim	Animation	data-parallel	coarse	large
ferret	Similarity Search	pipeline	medium	unbounded
fluidanimate	Animation	data-parallel	fine	large
freqmine	Data Mining	data-parallel	medium	unbounded
streamcluster	Data Mining	data-parallel	medium	medium
swaptions	Financial Analysis	data-parallel	coarse	medium
vips	Media Processing	data-parallel	coarse	medium
x264	Media Processing	pipeline	coarse	medium

TABLE A.3: The inherent key characteristics of PARSEC benchmarks [11].

Program/ Benchmarks	Data Usage	
	Sharing	Exchange
blackscholes, swaptions	low	low
bodytrack, freqmine	high	medium
canneal, dedup, ferret, x264	high	high
facesim, fluidanimate, streamcluster, vips	low	medium

TABLE A.4: The data usage behavior of PARSEC benchmarks [11].

A.2.1.1 Benchmark Descriptions

This section illustrates the properties of a few PARSEC benchmarks that are used to evaluate our proposed architecture designs. The details of the rest of the PARSEC benchmarks are reported in [11].

Bodytrack: The Bodytrack application records the 3D view of the human body through various cameras. The application uses an annealed practice filter to capture the 3D view using foreground and edge silhouette. The input video contains many frames which are used to select as a reference frame. The likelihood value is defined as a degree of the 3D body alignment with its foreground and its edges in the image frame. It is computed with the reference frame, and the frames are selected at different time-stamp. The value is calculated based on two attributes the foreground and the edge distance map. In the thread pool, the main thread assigns tasks to the other threads in the pool. Before proceeding further, the main thread has to wait for the remaining threads to complete their execution.

Fluidanimate: Fluidanimate is an Intel RMS application which is included in the PARSEC benchmark suite due to the increase in the physical simulation and real-time computer games animations. The application uses Smoothed Particle Hydrodynamic method [209]. There are five kernels used for modelling the incompressible fluids for interactive animation. The application produces output based on interpreting and discovering the surface of thick fluid.

Freqmine: Freqmine is an Intel RMS application developed by Concordia University and included in PARSEC benchmark due to the increasing demand for data mining techniques. The Freqmine application is used for array-based Frequent Itemset MIning (FIMI) [210]. The application is parallelised with OpenMP and uses three kernels.

Swaption: Swaption is also an Intel RMS workload that is used for pricing the portfolio by using the Heath-Jarrow-Morton (HJM) [211] method. The Swaption workload is added into the PARSEC benchmarks due to the increase in the value of Partial Differential Equation (PDE) and the Monte Carlo Simulation. Swaption stores all the portfolio in the swaption array and each entry of the array represents a derivative. The array is further divided into blocks equal to the number of spawned threads. Each block is assigned to a thread to ensure parallelism. The swaption application iterates through all the blocks and calls the module HJM Swaption blocking to compute a price.

Canneal: Canneal minimises the routing cost of chip design by using cache-aware Simulated Annealing (SA) technique. SA is a method used for large search space optimisation. The application chooses two pairs of elements randomly and swaps them. The algorithm discards only one element that effectively reduces the cache capacity misses during each iteration. Canneal represents the engineering workloads for fine-grained parallelism and lock-free synchronisation in the PARSEC

benchmark.

Dedup: The Dedup application compresses the data stream and communication data for future generation network systems. It uses global and local compression, called deduplication compression, to achieve a better compression ratio. The reason to include Dedup workload in the PARSEC is due to deduplication. The deduplication is the mainstream method to calculate storage footprint for the next-generation computing system.

Streamcluster: The Streamcluster application is used to solve the online clustering problem. The workload forms a different cluster of input point streams based on input median value. Each input point is assigned to the nearest centre to form a cluster. To measure the effectiveness of the clustering, it uses the sum of squared distance metric. The motive behind including Streamcluster in PARSEC is the increase in data mining algorithms and the predominant problem of streaming characteristics.

X264: The X264 application is an H.264/AVC (Advanced Video Coding) video encoder that adds new features in encoding. The key features are variable block-size motion compensation (VBSMC) or context-adaptive binary arithmetic coding (CBAC), high-resolution colour information, increased sample bit depth precision etc. The application generates high-quality encoding output with a low bit rate with increased encoding/decoding time. The application removes data redundancy by using a motion compensation technique. It is a flexible application with fulfilling demands of video conferencing, HD movie distribution etc.

Workload	Programming Language	Application Domain
400.perlbench	C	Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: Go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics / Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

TABLE A.5: The inherent key characteristics of CINT2006 benchmark suite [12]

A.2.2 SPEC CPU 2006

Standard Performance Evaluation Corporation (SPEC) CPU 2006 [12], is a CPU intensive industry-standardised benchmark suite. The SPEC applications are developed to emphasise the performance of the compiler, the computer processor (CPU) and the memory architecture. It includes two benchmark suites that concentrate on two different types of compute-intensive performance.

- **CINT2006 benchmark suite:** The CINT benchmark quantifies the compute-intensive integer performances. The suite contains twelve different benchmarks. The description of the workloads is given in table A.5.
- **CFP2006 benchmark suite:** The CFP benchmark quantify the compute-intensive floating-point performances. The suite contains seventeen different benchmark tests. The detailed description of the workloads is given in table A.6.

SPEC CPU 2006 measures compute-intensive performance of next-generation hardware by using real-world applications. The workloads can quantify computer performance in different ways, such as measuring the number of tasks completed by the computer in a definite time or measuring the speed of the computer to complete a single task.

Workload	Programming Language	Application Domain
410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics/Quantum Chromodynamics
434.zeusmp	Fortran	Physics / CFD
435.gromacs	C, Fortran	Biochemistry / Molecular Dynamics
436.cactusADM	C, Fortran	Physics / General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C, Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C, Fortran	Weather
482.sphinx3	C	Speech recognition

TABLE A.6: The inherent key characteristics of CFP2006 benchmark suite [12]

A.2.2.1 Benchmark Description

We characterise the properties of a few SPEC benchmarks that have been used in our work for the evaluation of different CMP based architectures. The detailed description of the rest of the benchmarks is reported in [12].

- **CINT2006 benchmarks**

1. **400.perlbench:** The workload is a partial version of Perl v5.8.7. It includes the email indexers: SpamAssassin and MHonArc and the tool specdiff that checks the benchmark output.
2. **401.bzip2:** The application is based on Julian Seward's bzip2 version 1.0.3. All the compression and decompression process in this benchmark is done entirely in memory, rather than I/O.
3. **403.gcc:** The workload is based upon GCC ver 3.2. The workload runs as a compiler with many optimisation flags enabled. It generates machine code for the AMD Opteron processor.
4. **429.mcf:** The benchmark is derived from MCF, a program used for vehicle scheduling in public mass transportation. It exploits a simple network algorithm to schedule public transport.
5. **445.gobmk:** The program plays an artificial game: Go, a simple-looking but deep complex inside.

6. **456.hmmer:** The workload is used in computational biology to search DNA sequence pattern. The application uses statistical hidden Markov model of multiple sequence alignment.
7. **458.sjeng:** The workload is based on the program Sjeng ver. 11.2 that plays chess and a variety of chess variants like losing chess and drop-chess.
8. **462.libquantum:** The workload models a quantum computer that is based on quantum mechanics and solves real hard tasks in polynomial time. To facilitate this process, it uses Shor's polynomial-time factorisation algorithm.
9. **464.h264ref:** The workload is an implementation of H.264/AVC coding technique that is expected to replace MPEG2.
10. **471.omnetpp:** The workload models a vast ethernet network using discrete event simulation.
11. **473.astar:** The workload is derived from the well known 2D-path finding libraries used in AI games. It models different variants of A* path-finding algorithms based upon the requirement.

- **CFP2006 benchmarks**

1. **410.bwaves:** The workload models the blast wave as a three dimensional transonic transient laminar viscous flow.
2. **416.gamess:** The workload models the different varieties of quantum chemical computations. It performs the self-consistent field calculations using Multi-Configuration Self-Consistent Field, Restricted Hartree Fock method, and Restricted open-shell Hartree-Fock.
3. **433.milc:** The workload models the four-dimensional SU(3) lattice gauge theory using Von-Neumann MIMD parallel machines.
4. **434.zeusmp:** The workload models the astrophysical phenomena. The application resolves the problems in three spatial dimensions with a wide variety of boundary constraints.

5. **435.gromacs:** It is used to perform molecular dynamics. It models the Newtonian equations of motion for systems with hundreds to millions of particles.
6. **436.cactusADM:** The workload is a combination of Cactus, an open-source problem-solving environment, and BenchADM, kernel representative of numerical relativity. The application solves the Einstein evolution equation using leapfrog numerical method.
7. **437.leslie3d:** The workload is based on LESlie3d (Large-Eddy Simulations with Linear-Eddy Model in 3D). It solves the problem of Computational Fluid Dynamics (CFD) using MacCormack predictor-corrector time integration scheme.
8. **444.namd:** The workload models the large bio-molecular systems. It tests the atoms of apolipoprotein A-I.
9. **447.dealII:** The workload is based on dealII, a library targeted at adaptive finite elements and error estimation. The application provides a solution for the Helmholtz-type equation with non-constant coefficients.
10. **450.soplex:** The application provides a solution for the linear program using a simplex algorithm and sparse linear algebra.
11. **454.calculix:** The workload is derived from CalculiX, finite element code for linear and nonlinear three-dimensional structural application. The application provides the solution for buckling, eigenmode analysis, etc.
12. **465.tonto:** The workload is an open-source quantum chemistry package. It performs the calculation of Hartree-Fock wave function to match experimental X-ray diffraction data.
13. **470.lbm:** The workload implements Lattice-Boltzmann Method to model incompressible fluid in 3D.

A.3 Simulation Procedure

In this dissertation, we have used several multi-threaded and multi-programmed benchmarks for the simulation analysis of the proposed techniques.

A.3.1 Multi-threaded vs Multi-programmed Workloads

The PARSEC benchmarks are multi-threaded workload where the number of threads in each program depends upon the input size and load of the program. The number of threads is identified as a command-line argument in the majority of the benchmarks. During the execution, the multi-threading occurs in a specific period called Region Of Interest (ROI). In other words, the real PARSEC application executes in ROI. The input scanning, variable initialisation etc. happens before ROI. Once ROI executes, the workload generates output and then terminates.

Multiple SPEC CPU 2006 workloads are merged to build multi-programmed benchmarks using M5 commands in the target system. The different processes execute on the different cores until any one process completes the prespecified number of instructions. In multi-programmed benchmarks, the phrase benchmark represents the combined workload mix.

A.3.2 Benchmarks Used in Our Simulations

Different mixtures of the workloads can be made based on the multi-threaded and multi-programmed workloads provided by PARSEC and SPEC CPU 2006. These mixtures can be either of single PARSEC benchmark application with multiple threads or a mix of different processes from the SPEC CPU 2006. In our work, we have used a different combination of workloads for the proposed techniques. Each chapter has specific workloads to the proposed techniques in the chapter. The details of the benchmarks used in our simulation analysis are given in the Result and Analysis section of Chapters 3, 4, 5 and 6.

A.3.3 Benchmark Running Process

The PARSEC workloads are run on the target machine until the completion of the workloads. Here, the running process has four stats dumped in the generated stats file. These stats are (a) statistics for M5 full system booting process, (b) statistic before reaching ROI that include the initialisation of benchmarks and the spawning of the threads, (c) the statistics in the ROI and (d) the statistics from the end of the ROI to the simulation exit. We focus on the third stat, ROI, which represents the execution of the application.

To run the multi-programmed benchmarks, the very first step is to load all the applications serially. To warm-up, each application is executed for one billion instructions. The warm-up phase is necessary to go beyond the compulsory misses in the cache. It allows the proposed architecture to settle properly in the simulator. After warming-up, each workload is run for one million instructions to collect the required stats needed to analyse the performance of the proposed design.

A.3.4 Comparing Different CMP Architectures

There are several performance parameters such as energy consumption, EDP, lifetime, write variation, IPC, implementation overhead, etc., to calculate the effectiveness of the proposed CMP architecture with other existing architectures. All our proposed and prior designs are implemented on GEM5 (full system simulator) to facilitate a fair comparison of proposed techniques with the existing designs. We execute various PARSEC and SPEC CPU 2006 application on top of the implemented architectures in GEM5. Many varieties of statistics are recorded during the execution of each workload, as reported in section A.1.2.1. The efficacy of architectures is evaluated based on the generated stats.

In general, the architecture is engineered with different design configurations, such as different cache sizes, various VC depths, different number of VCs per VNet etc. As per need, the appropriate details are provided in the relevant chapters/sections. The process of executing benchmarks is kept unchanged to maintain regularity

for all the architectures with different configurations. We have illustrated each workload result separately, and the geometric mean of all benchmarks are derived in our result sections.

A.4 Hardware Synthesis of Zero Detection Logic

Xilinx ISE [5] [212] is a software tool from Xilinx for synthesis and analysis of HDL (High Definition Language) designs. The Xilinx ISE is primarily used for circuit synthesis and design. It is an integrated synthesis environment which primarily targets development of embedded firmware for Xilinx FPGA and CPLD integrated circuit (IC) product families. There several components of ISE such as Embedded Development Kit (EDK), a Software Development Kit (SDK) and ChipScope Pro.

The major features of ISE are to synthesize/compile new designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. The design flow of Xilinx ISE includes compilation, execution (simulation), debugging, RTL implementation, synthesis report generation about design metrics, RTL verification and RTL implementation packaging into supported IP formats.

Project Navigator is the primary user interface of ISE. This includes **design hierarchy**, **source code editor**, an **output console** and **process tree** [212]. **Design hierarchy** contains design modules which are displayed as a tree structure and the dependencies of these modules are interpreted by ISE [212]. A design may have one or more modules in hierarchy. The module also includes the configuration of the design constraints and, pin configuration and mapping. The **process tree** describes the operation performed by ISE on active module. The tree has compliation functions, module dependency functions, etc. The issues related to the current running operations are informed and reported as errors and warnings.

The system level test programs written in HDL languages, are simulated with either ISIM or ModelSim logic simulator [213]. The test bench programs may

contain simulated input signal waveforms or monitors which observe and verify the outputs of the device under test. The simulators can perform different types of simulations such as **logical verification**, **behavioural verification** and **post-place & route simulation**. The **logical verification** ensures that the module produces expected results; whereas, the behavioural verification verifies the logical and timing issues. The post-plane verification is to verify the behaviour after module is placed within the reconfigurable logic of the FPGA.

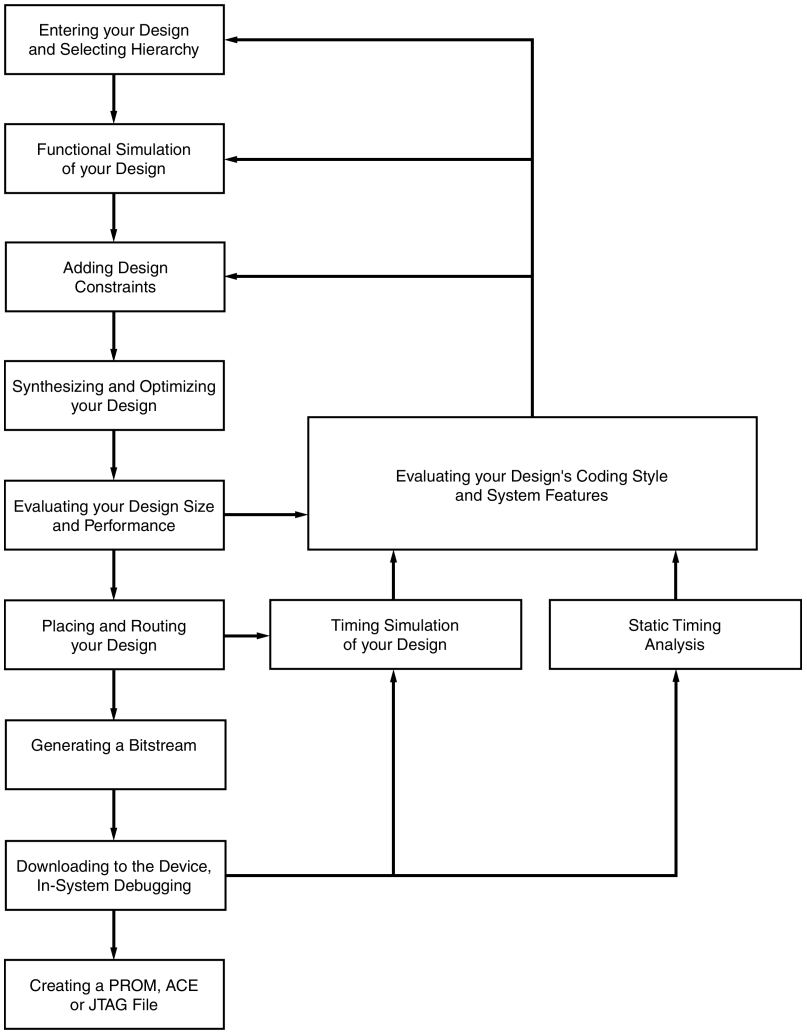


FIGURE A.2: Design flow diagram [5]

Xilinx synthesis algorithm is up to 30% fater than other competing programs which allows bigger logic density with reduced project time and costs [214]. For the more complex designs, such as complex memory blocks and I/O blocks of FPGA, more complex synthesis algorithms are used which separate unrelated modules into slices

and reduce post-placement errors. The FPGA synthesis flowchart is given in figure A.2. Xilinx and other third-party vendors offered IP Cores to implement system-level functions such as digital signal processing (DSP), bus interfaces, networking protocols, image processing, embedded processors, and peripherals. Xilinx has been used widely to implement ASIC-based design to FPGA-based designs.

ISE supports Xilinx's 7-series (except of Spartan-7) and older devices including CPLDs (XC9500 and CoolRunner) [215]. Xilinx Vivado supports newer version of Xilinx devices. The operating supported by Xilinx are Windows, Red Hat Enterprise 4, 5, & 6 Workstations (32 & 64 bits) and SUSE Linux Enterprise 11 (32 & 64 bits). Other GNU/Linux distributions can run Xilinx ISE WebPack with some modifications or configurations [216].

A.4.1 Synthesis Report

In this section we present the synthesis report of Zero Detection Logic (ZDL) generated by Xilinx ISE tool. The synthesis report includes the synthesis options summary, HDL Parsing, HDL Elaboration, HDL Synthesis, Advanced HDL Synthesis, Low Level Synthesis, Partition Report and Design Summary. The synthesis options summary lists source and target parameters and general options.

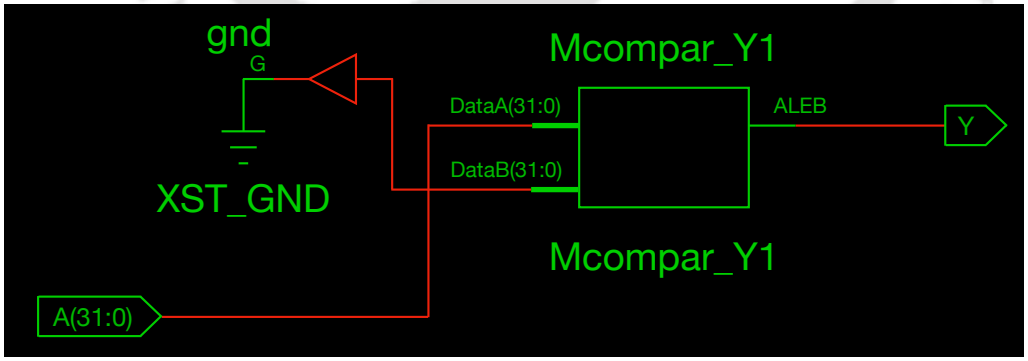


FIGURE A.3: RTL schematic view of Zero Detection Logic.

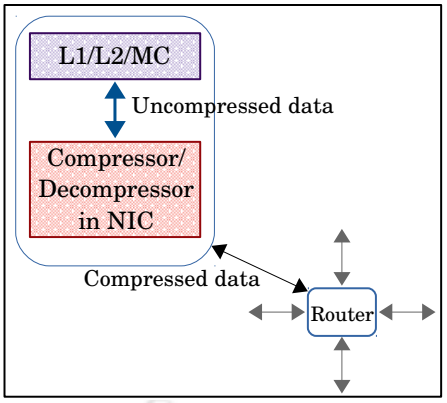


FIGURE A.4: Network interface controller compression schematic

A.4.1.1 Design Summary

Figure A.4 shows the network interface controller where the compressor and decompressor fit. Here, at the source NI, the data packet is compressed before flitisization and injected into the network. On the other side, at the receiving NI, the flits are decompressed before packetized and sent to protocol buffer. ZDL is the main component of the compressor. The ZDL check for all words and outputs 1 if the word is a zero-word, 0 otherwise. The output of ZDL gets stored in *isZeroVector* which is used at the destination NI for the decompression of the packet. Note that the compressor/ decompressor resides at the NI, and the compression does not affect the router cycles.

The design summary reports the timing summary and device utilization summary for the target device. The number and type of components that constitute the required circuit's datapath synthesis are 1 GND (Ground), 1 LUT2, 6 LUT5, 7 MUXCY (Multiplexer), and 1 VCC. Whereas the total number of Input and Output buffers used in the synthesis circuit is 33, of which 32 are input buffers, and 1 is output buffer. On the other hand, the timing delay of different essential components is Input Buffer (1.328 ns), LUT5 (0.254 ns), one MUX (0.215 ns), five MUXes (0.023 ns), and 1 MUX (0.235). Hence, the total delay required for the execution is 3.942 ns. Out of which, 2.149 ns is spent on logic realization (73.8%), and 1.793ns (26.2%) is spent on route realization. The timing summary gives the time consumed fanout of each gate in the design. The compiled design of ZDL A.3

uses 32-bit input and 1-bit output. The output is one if all the input bits are zero, 0 otherwise. The gate delay for the input buffer is 1.328ns, which is the maximum among all components in the design.

A.5 Analytical modelling of latency degradation of STT-RAM-based NoC

We use generalized analytical router model discussed in [217] as a base. In [217], authors model a single router as a set of first-come-first-serve buffers connected by a crossbar switch. Since a single packet gets allocated to a virtual channel, the average size of a buffer for an input channel can be represented as the *average number of packets* at the input buffers. This can also be used to calculate *average waiting time* in buffers, which is given by [217] :

$$W_{ij} = \frac{N_{ij}}{\lambda_{ij}} \tag{A.7}$$

where W_{ij} is average waiting time, N_{ij} is average buffer utilization (in terms of flits) and λ_{ij} is packet arrival rate at the channel j buffer at router i.

The *average packet latency* at the router can be calculated using average waiting time and average packet service time (T) at the router. A delay at each router contributes the packet latency from source to destination, which can be used to compute average packet latency in the network. When a packet is sent from source s to destination d, it traverses a set of routers and the corresponding input buffers denoted by Π_{sd}. The *average latency* for a packet from source s to destination d is given by [217] :

$$L_{sd} = W_s + \sum_{(i,j)\in\Pi_{sd}} (W_{ij} + T) \tag{A.8}$$

where W_s is the queueing delay at the source, W_{ij} is the average waiting time at the channel j of router i, and T is the average service time.

With combination of equation A.8 and packet transmission rate (x_{sd}) from node s to node d, the *overall average packet latency* for a network is given by [217] :

$$L = \frac{\sum_{\forall s,d} x_{sd} \times L_{sd}}{\sum_{\forall s,d} x_{sd}} \tag{A.9}$$

We use above formalisms to model the STTRAM based NoC router. There are two router designs with STT-RAM and SRAM based buffers. As we have shown in previous sections that the STT-RAM buffer takes more cycles to write in comparison to SRAM based buffers, and this impacts the average packet latency of the network. L^{STT} and L^{SRAM} represent the average packet latency in STT-RAM and SRAM based router design, respectively. We denoted latency degradation by Δ, which can be written as:

$$L^{STT} = L^{SRAM} + \Delta \tag{A.10}$$

The slow STT adds delay to the packet transmission at each router, hence the L_{sd} for a packet in STT and SRAM based design will have same relation as equation A.10, which can be written as:

$$L_{sd}^{STT} = L_{sd}^{SRAM} + \Delta_{sd} \tag{A.11}$$

We assume that the packet generation rate at the *PE* and the queueing latency at the source is same in STT and SRAM design. The latency degradation Δ will only depend on the network latency of each packet. With this assumptionand equation A.8, we can re-write equation A.11 as:

$$\sum_{(i,j)\in\Pi_{sd}} (W_{ij}^{STT} + T^{STT}) = \sum_{(i,j)\in\Pi_{sd}} (W_{ij}^{SRAM} + T^{SRAM}) + \Delta_{sd} \tag{A.12}$$

where W_{ij}^{STT} and W_{ij}^{SRAM} are the average waiting time in STT and SRAM design respectively. T^{STT} denotes the average service time in STT router and T^{SRAM} is the average service time in SRAM router.

From A.7 and A.12,

$$\sum_{(i,j)\in\Pi_{sd}} (\frac{N_{ij}^{STT}}{\lambda_{ij}^{STT}} + T^{STT}) = \sum_{(i,j)\in\Pi_{sd}} (\frac{N_{ij}^{SRAM}}{\lambda_{ij}^{SRAM}} + T^{SRAM}) + \Delta_{sd} \tag{A.13}$$

Let us say, there are n routers on the path from s to d and the buffer utilization is the same for each router.

$$N_{ij}^{STT} = N^{STT}, \forall\ i \in \Pi_{sd} \tag{A.14}$$

$$N_{ij}^{SRAM} = N^{SRAM}, \forall\ i \in \Pi_{sd} \tag{A.15}$$

In both designs, STT and SRAM, the traffic arrival rate (λ_{ij}) at the source is assumed to be same. However, the packet arrival rate of STT routers from s to d will reduce due to the delayed service time at each hop. For the sake of simplicity we assume that the packet arrival rate is same at intermediate routers from s to d in a design ($\lambda_{ij}^{STT} = \lambda_{ik}^{STT} = ... = \lambda_{id}^{STT}$). Now if we assume the packet arrival rate reduces by α in STT design, we can write ,

$$\lambda_{ij}^{STT} + \alpha = \lambda_{ij}^{SRAM} = \lambda \tag{A.16}$$

Using equation A.14, A.15 and A.16, we can write A.13 as:

$$n \times T^{STT} + n \times \frac{N^{STT}}{\lambda - \alpha} = n \times T^{SRAM} + n \times \frac{N^{SRAM}}{\lambda} + \Delta_{sd} \tag{A.17}$$

$$\implies \Delta_{sd} = n \times (\frac{N^{STT}}{\lambda - \alpha} - \frac{N^{SRAM}}{\lambda}) + n \times (T^{STT} - T^{SRAM}) \tag{A.18}$$

If the buffer utilization for each design is same (i.e. $N^{STT} = N^{SRAM} = N$), then

$$\Delta_{sd} = n \times [\frac{\alpha N}{\lambda(\lambda - \alpha)} + (T^{STT} - T^{SRAM})] \tag{A.19}$$

If we extract the relation of α with λ and n, we get the following equations:

$$\alpha = \frac{C}{n} \tag{A.20}$$

where C is some constant.

$$\alpha = \frac{C_1\lambda^2}{C_2\lambda + C_3} \tag{A.21}$$

where C_1, C_2 and C_3 are some constant.

From A.19, the packet transmission rate reduction in STT (α) is inversely proportional to the number of routers between source s to destination d. In other words, as the distance between the source and destination increases the value of α will reduce with the hop count.

We have done an analysis for the proposed modelling using synthetic traffic pattern on Garnet2.0 standalone simulation. The standalone analysis was done for 4×4 mesh network with XY routing and 3 stage pipeline router design. The simulation was done for fixed source and destination pairs, hence Δ_{sd} becomes Δ.

On our experimental analysis, with different inject rates and hop-counts, we have observed the same behaviour of α (as above) which is tabulated in Table A.7. The smaller value of hop count has a smaller value of α, which introduces a small delay to packet arrival rate in STT routers and hence the latency difference of STT and SRAM design is less. The higher injection rate increases the average waiting time at each intermediate router. This results in an increased value of α and hence more latency degradation in STT design.

From A.19 we see that the latency degradation (Δ) is directly proportional to the number of hops (n). The degradation, $T^{STT} - T^{SRAM}$, accumulates at each hop. The difference between T^{STT} and T^{SRAM} is 1 cycle as STT takes 1 extra cycle to

Hop Count (n) = 7				**Injection Rate (λ) = 0.05**			
Inj. Rate (λ)	**0.01**	**0.05**	**0.09**	**Hop Count (n)**	**3**	**5**	**7**
Latency deg.	8	8.4	11.2	**Latency deg.**	4.3	6.38	8.4
α	0.00005	0.0017	0.014	α	0.00157	0.00166	0.0017

TABLE A.7: Relation of α with hop count and injection rate.

get serviced due to its slow writing speed. The first term of A.19 contributes to the small increase in latency on account of the injection rate (λ) and hop count (n). As observed from earlier analysis, the value of the first term depends on the injection rate (λ) and hop-count (n). In particular, the value is more for higher injection rate, and the value gradually reduces with increased hop-distance. For a given SRAM router design, equation A.13 can be used to model STT-RAM router with a fixed latency degradation.

In a case study, we kept injection rate fixed to 0.05 packets/cycle and changed hop-counts. With hop-count 3 Δ obtained using equation A.19 is $3 + \epsilon$ and using simulation it is 4.3. With hop-count 5 Δ obtained using equation A.19 is $5 + \epsilon$ and using simulation it is 6.38. In the absence of ϵ, theory Δ is almost close to simulated Δ value.

Bibliography

[1] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, "Hybrid Drowsy SRAM and STT-RAM Buffer Designs for Dark-Silicon-Aware NoC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3041–3054, 2016.

[2] D. Zoni, A. Canidio, W. Fornaciari, P. Englezakis, C. Nicopoulos, and Y. Sazeides, "Blackout: Enabling fine-grained power gating of buffers in network-on-chip routers," *Journal of Parallel and Distributed Computing*, vol. 104, pp. 130–145, 2017.

[3] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 508–519.

[4] "Transistor count statistics," https://www.ncbi.nlm.nih.gov, 2019.

[5] "ISE Design Suite 14.4 through 14.7." [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/sim.pdf

[6] M. H. Kryder and C. S. Kim, "After hard drives—What comes next?" *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 3406–3413, 2009.

[7] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, "Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement," in *2008 45th ACM/IEEE Design Automation Conference*. IEEE, 2008, pp. 554–559.

[8] S. Mittal, J. S. Vetter, and D. Li, "A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1524–1537, June 2015.

[9] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of Candidate Device Technologies for Storage-Class Memory," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 449–464, July 2008.

[10] D. L. Lewis and H. S. Lee, "Architectural Evaluation of 3D Stacked RRAM Caches," in *2009 IEEE International Conference on 3D System Integration*, Sep. 2009, pp. 1–4.

[11] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, Jan. 2011. [Online]. Available: http://parsec.cs.princeton.edu/

[12] J. L. Henning, "SPEC CPU2006 Benchmark Descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, sept 2006.

[13] G. Moore, "Moore's law," *Electronics Magazine*, vol. 38, no. 8, p. 114, 1965.

[14] G. E. Moore, "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[15] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[16] [Online]. Available: http://www.intel.com

[17] [Online]. Available: https://www.oracle.com/sun/index.html

[18] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-Way Multithreaded Sparc Processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, March 2005.

[19] M. Palesi and M. Daneshtalab, *Routing algorithms in networks-on-chip*. Springer, 2014.

[20] N. E. Jerger and L. S. Peh, *On-chip Networks*, 1st ed. Morgan, 2009.

[21] W. Zang and A. Gordon-Ross, "A Survey on Cache Tuning from a Power/Energy Perspective," *ACM Computing Surveys*, vol. 45, no. 3, Jun. 2013.

[22] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2009, pp. 469–480.

[23] S. Kaxiras and M. Martonosi, *COMPUTER ARCHITECTURE TECHNIQUES FOR POWER-EFFICIENCY*. Mark D. Hill, University of Wisconsin, Madison, 2008.

[24] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[25] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proceedings of the conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 423–428.

[26] K. Roy and S. C. Prasad, *Low-Power Cmos Vlsi Circuit Design*. John Wiley and Sons, 2009.

[27] Y.-C. Yeo, T.-J. King, and C. Hu, "Mosfet gate leakage modeling and selection guide for alternative gate dielectrics based on leakage considerations," *IEEE Transactions on Electron Devices*, vol. 50, no. 4, pp. 1027–1035, April 2003.

[28] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Maximizing performance of thermally constrained multi-core processors by dynamic voltage and frequency control," in *IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 310–313.

[29] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[30] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual international symposium on computer architecture (ISCA)*. IEEE, 2011, pp. 365–376.

[31] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.

[32] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward Dark Silicon in Servers," *IEEE Micro*, vol. 31, no. 4, pp. 6–15, July 2011.

[33] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May 2012.

[34] "International Technology Roadmap for Semiconductors. The Model for Assessment of CMOS Technologies and Roadmaps (MASTAR)," http://www.itrs.net/models.html, [Online].

[35] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The eda challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.

[36] M. Shafique, S. Garg, T. Mitra, S. Parameswaran, and J. Henkel, "Dark Silicon as a Challenge for Hardware/Software Co-Design," in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2014, pp. 1–10.

[37] J. Henkel, H. Bukhari, S. Garg, M. U. K. Khan, H. Khdr, F. Kriebel, U. Ogras, S. Parameswaran, and M. Shafique, "Dark silicon: From computation to communication," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 23.

[38] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.* IEEE, 2003, pp. 105–116.

[39] H.-S. Wang, L.-S. Peh, and S. Malik, "A power model for routers: Modeling alpha 21364 and infiniband routers," *IEEE Micro*, vol. 23, no. 1, pp. 26–35, 2003.

[40] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[41] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee *et al.*, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose programs," *IEEE micro*, vol. 22, no. 2, pp. 25–35, 2002.

[42] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown *et al.*, "Tile64-Processor: A 64-Core SOC with Mesh Interconnect," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International.* IEEE, 2008, pp. 88–598.

[43] D. Apalkov, A. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. Lottis, K. Moon, X. Luo, E. Chen, A. Ong, A. Driskill-Smith, and M. Krounbi,

"Spin-transfer Torque Magnetic Random Access Memory (STT-MRAM)," *J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 13:1–13:35, May 2013.

[44] H. Jang, B. S. An, N. Kulkarni, K. H. Yum, and E. J. Kim, "A Hybrid Buffer Design with STT-MRAM for On-Chip Interconnects," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*. IEEE, 2012, pp. 193–200.

[45] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto *et al.*, "A novel NonVolatile Memory with Spin Torque Transfer Magnetization Switching: Spin-RAM," in *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*. IEEE, 2005, pp. 459–462.

[46] O. Golonzka, J.-G. Alzate, U. Arslan, M. Bohr, P. Bai, J. Brockman, B. Buford, C. Connor, N. Das, B. Doyle *et al.*, "MRAM as Embedded Non-Volatile Memory Solution for 22FFL FinFET Technology," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 18–1.

[47] Y. Song, J. Lee, S. Han, H. Shin, K. Lee, K. Suh, D. Jeong, G. Koh, S. Oh, J. Park *et al.*, "Demonstration of Highly Manufacturable STT-MRAM Embedded in 28nm Logic," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 18–2.

[48] C. Li and P. Ampadu, "A compact low-power edram-based noc buffer," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2015, pp. 116–121.

[49] (2014). [Online]. Available:, https://www.mram-info.com/companies, 2014.

[50] ——, http://www.memorystrategies.com/report/embeddeddram.html, 2014.

[51] ——, http://www.economist.com/node/21560981, 2014.

[52] (2020). [Online]. Available:, https://www.anandtech.com/show/15557/globalfoundries-22fdx-mram, 2020.

[53] (2016). [Online]. Available:, https://www.invecas.com/press-releases/globalfoundries-launches-embedded-mram-on-22fdx-platform/, 2016.

[54] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology Comparison for Large Last-Level Caches (L 3 Cs): Low-leakage SRAM, Low Write-Energy STT-RAM, and Refresh-Optimized eDRAM," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 143–154.

[55] Y. Huai *et al.*, "Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects," *AAPPS bulletin*, vol. 18, no. 6, pp. 33–40, 2008.

[56] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache Revive: Architecting Volatile STT-RAM Caches for Enhanced Performance in CMPs," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 243–252.

[57] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*. IEEE, 2011, pp. 50–61.

[58] H. Li and Y. Chen, "An Overview of Non-Volatile Memory Technology and the Implication for Tools and Architectures," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 731–736.

[59] S. . Chung, T. Kishi, J. W. Park, M. Yoshikawa, K. S. Park, T. Nagase, K. Sunouchi, H. Kanaya, G. C. Kim, K. Noma, M. S. Lee, A. Yamamoto, K. M. Rho, K. Tsuchida, S. J. Chung, J. Y. Yi, H. S. Kim, Y. S. Chun, H. Oyamatsu, and S. J. Hong, "4Gbit Density STT-MRAM using Perpendicular MTJ realized with Compact Cell Structure," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec 2016, pp. 27.1.1–27.1.4.

[60] Y. Lu, T. Zhong, W. Hsu, S. Kim, X. Lu, J. J. Kan, C. Park, W. C. Chen, X. Li, X. Zhu, P. Wang, M. Gottwald, J. Fatehi, L. Seward, J. P. Kim,

N. Yu, G. Jan, J. Haq, S. Le, Y. J. Wang, L. Thomas, J. Zhu, H. Liu, Y. J. Lee, R. Y. Tong, K. Pi, D. Shen, R. He, Z. Teng, V. Lam, R. Annapragada, T. Torng, P. Wang, and S. H. Kang, "Fully functional perpendicular stt-mram macro embedded in 40 nm logic for energy-efficient iot applications," in *2015 IEEE International Electron Devices Meeting (IEDM)*, Dec 2015, pp. 26.1.1–26.1.4.

[61] O. Golonzka, J. . Alzate, U. Arslan, M. Bohr, P. Bai, J. Brockman, B. Buford, C. Connor, N. Das, B. Doyle, T. Ghani, F. Hamzaoglu, P. Heil, P. Hentges, R. Jahan, D. Kencke, B. Lin, M. Lu, M. Mainuddin, M. Meterelliyoz, P. Nguyen, D. Nikonov, K. O'brien, J. O. Donnell, K. Oguz, D. Ouellette, J. Park, J. Pellegren, C. Puls, P. Quintero, T. Rahman, A. Romang, M. Sekhar, A. Selarka, M. Seth, A. J. Smith, A. K. Smith, L. Wei, C. Wiegand, Z. Zhang, and K. Fischer, "MRAM as Embedded Non-Volatile Memory Solution for 22FFL FinFET Technology," in *2018 IEEE International Electron Devices Meeting (IEDM)*, Dec 2018, pp. 18.1.1–18.1.4.

[62] Y. J. Song, J. H. Lee, S. H. Han, H. C. Shin, K. H. Lee, K. Suh, D. E. Jeong, G. H. Koh, S. C. Oh, J. H. Park, S. O. Park, B. J. Bae, O. I. Kwon, K. H. Hwang, B. Y. Seo, Y. K. Lee, S. H. Hwang, D. S. Lee, Y. Ji, K. C. Park, G. T. Jeong, H. S. Hong, K. P. Lee, H. K. Kang, and E. S. Jung, "Demonstration of Highly Manufacturable STT-MRAM Embedded in 28nm Logic," in *2018 IEEE International Electron Devices Meeting (IEDM)*, Dec 2018, pp. 18.2.1–18.2.4.

[63] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. . Chen, R. M. Shelby, M. Salinga, D. Krebs, S. . Chen, H. . Lung, and C. H. Lam, "Phase-Change Random Access Memory: A Scalable Technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 465–479, July 2008.

[64] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 2–13.

[65] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 14–23.

[66] K. Lee, B. Cho, W. Cho, S. Kang, B. Choi, H. Oh, C. Lee, H. Kim, J. Park, Q. Wang, M. Park, Y. Ro, J. Choi, K. Kim, Y. Kim, I. Shin, K. Lim, H. Cho, C. Choi, W. Chung, D. Kim, K. Yu, G. Jeong, H. Jeong, C. Kwak, C. Kim, and K. Kim, "A 90nm 1.8V 512Mb Diode-Switch PRAM with 266MB/s Read Throughput," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Feb 2007, pp. 472–616.

[67] Y. Choi, I. Song, M. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y. Lee, J. Yoo, and G. Jeong, "A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth," in *2012 IEEE International Solid-State Circuits Conference*, Feb 2012, pp. 46–48.

[68] H. Akinaga and H. Shima, "Resistive Random Access Memory (ReRAM) Based on Metal Oxides," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2237–2251, Dec 2010.

[69] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive Switching Mechanism for Metal/Oxide/Metal Nanodevices," *Nature nanotechnology*, vol. 3, no. 7, p. 429, 2008.

[70] Y. Ho, G. M. Huang, and P. Li, "Non-Volatile Memristor Memory: Device Characteristics and Design Implications," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, Nov 2009, pp. 485–490.

[71] R. S. Williams, "How We Found The Missing Memristor," *IEEE Spectrum*, vol. 45, no. 12, pp. 28–35, Dec 2008.

[72] "What Happened To ReRAM?" [Online]. Available: https://semiengineering.com/what-happened-to-reram/

[73] J. Ahn, S. Yoo, and K. Choi, "Write Intensity Prediction for Energy-Efficient Non-Volatile Caches," in *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2013, pp. 223–228.

[74] ——, "Prediction Hybrid Cache: An Energy-Efficient STT-RAM Cache Architecture," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 940–951, 2015.

[75] H. Jang, R. Boyapati, R. Kansal, K. H. Yum, and E. J. Kim, "A multibank buffer design with stt-mram for high-bandwidth low-power on-chip interconnects."

[76] J. Wang, X. Dong, Y. Xie, and N. P. Jouppi, "i2WAP: Improving Non-Volatile Cache Lifetime by reducing Inter- and Intra-set Write Variations," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 234–245.

[77] S. Mittal and J. S. Vetter, "EqualChance: Addressing Intra-set Write Variation to Increase Lifetime of Non-volatile Caches," in *2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW 14)*. Broomfield, CO: USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/inflow14/workshop-program/presentation/mittal

[78] S. Mittal, J. S. Vetter, and D. Li, "LastingNVCache: A Technique for Improving the Lifetime of Non-volatile Caches," in *2014 IEEE Computer Society Annual Symposium on VLSI*, July 2014, pp. 534–540.

[79] S. Mittal, J. S. Vetter, and D. Li, "WriteSmoothing: Improving Lifetime of Non-volatile Caches Using Intra-set Wear-leveling," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '14. New York, NY, USA: ACM, 2014, pp. 139–144.

[80] S. Mittal and J. Vetter, "A Technique for Improving Lifetime of Non-Volatile Caches Using Write-Minimization," *Journal of Low Power Electronics and Applications*, vol. 6, no. 1, 2016. [Online]. Available: https://www.mdpi.com/2079-9268/6/1/1

[81] M. R. Jokar, M. Arjomand, and H. Sarbazi-Azad, "Sequoia: A High-Endurance NVM-Based Cache Architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 954–967, March 2016.

[82] M. Soltani, M. Ebrahimi, and Z. Navabi, "Prolonging Lifetime of Non-Volatile Last Level Caches with Cluster Mapping," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, May 2016, pp. 329–334.

[83] T. Majumder, M. Suri, and V. Shekhar, "Noc router using stt-mram based hybrid buffers with error correction and limited flit retransmission," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 2305–2308.

[84] N. Wang, R. Boyapati, J. Huang, V. kumar Ilanchelian, K. H. Yum, and E. J. Kim, "Fly-over: A light-weight distributed power-gating mechanism for energy-efficient networks-on-chip."

[85] H. Farrokhbakht, H. M. Kamali, N. E. Jerger, and S. Hessabi, "Sponge: A scalable pivot-based on/off gating engine for reducing static power in noc routers," in *Proceedings of the International Symposium on Low Power Electronics and Design*. ACM, 2018, p. 17.

[86] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "Ddrnoc: Dual data-rate network-on-chip," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 2, pp. 1–24, 2018.

[87] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Shortpath: A network-on-chip router with fine-grained pipeline bypassing," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3136–3147, 2016.

[88] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "Freewaynoc: a ddr noc with pipeline bypassing," in *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2018, pp. 1–8.

[89] H. Farrokhbakht, H. M. Kamali, and N. E. Jerger, "Muffin: Minimally-buffered zero-delay power-gating technique in on-chip routers," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2019, pp. 1–6.

[90] H. Farrokhbakht, M. Taram, B. Khaleghi, and S. Hessabi, "Toot: an efficient and scalable power-gating method for noc routers," in *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, 2016, pp. 1–8.

[91] L. Chen and T. M. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2012, pp. 270–281.

[92] T. Krishna, C.-H. O. Chen, W.-C. Kwon, and L.-S. Peh, "Smart: single-cycle multihop traversals over a shared network on chip," *IEEE micro*, vol. 34, no. 3, pp. 43–56, 2014.

[93] I. Pérez, E. Vallejo, and R. Beivide, "Smart++ reducing cost and improving efficiency of multi-hop bypass in noc routers," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–8.

[94] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh, "Breaking the on-chip latency barrier using smart," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 378–389.

[95] I. Pérez, E. Vallejo, and R. Beivide, "Efficient router bypass via hybrid flow control," in *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*. IEEE, 2018, pp. 1–6.

[96] L. Wang, S. Ma, C. Li, W. Chen, and Z. Wang, "A high performance reliable noc router," *Integration*, vol. 58, pp. 583–592, 2017.

[97] T. Krishna, C.-H. O. Chen, S. Park, W.-C. Kwon, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "Single-cycle multihop asynchronous repeated traversal: A smart future for reconfigurable on-chip networks," *Computer*, vol. 46, no. 10, pp. 48–55, 2013.

[98] T. Krishna and L.-S. Peh, "Single-cycle collective communication over a shared network fabric," in *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*. IEEE, 2014, pp. 1–8.

[99] P. Poluri and A. Louri, "Shield: A reliable network-on-chip router architecture for chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 3058–3070, 2016.

[100] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "dark-noc: Designing energy-efficient network-on-chip with multi-vt cells for dark silicon," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–6.

[101] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks." in *HPCA*, vol. 3, 2003, pp. 91–102.

[102] V. Soteriou and L.-S. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks," *IEEE Transactions on Parallel & Distributed Systems*, no. 3, pp. 393–408, 2007.

[103] J. M. Stine, N. P. Carter, and J. Flich, "Comparing adaptive routing and dynamic voltage scaling for link power reduction," *IEEE Computer Architecture Letters*, vol. 3, no. 1, pp. 4–4, 2004.

[104] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Adding slow-silent virtual channels for low-power on-chip networks," in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2008, pp. 23–32.

[105] M. Kar and T. Krishna, "A case for low frequency single cycle multi hop nocs for energy efficiency and high performance," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 743–750.

[106] R. Hesse and N. E. Jerger, "Improving dvfs in nocs with coherence prediction," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, 2015, pp. 1–8.

[107] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "Malleable noc: Dark silicon inspired adaptable network-on-chip," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 1245–1248.

[108] J. Postman, T. Krishna, C. Edmonds, L.-S. Peh, and P. Chiang, "Swift: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1432–1446, 2012.

[109] J.-J. Han, M. Lin, D. Zhu, and L. T. Yang, "Contention-aware energy management scheme for noc-based multicore real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 691–701, 2014.

[110] M. R. Casu and P. Giaccone, "Rate-based vs delay-based control for dvfs in noc," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1096–1101.

[111] Y. Yao and Z. Lu, "Dvfs for nocs in cmps: A thread voting approach," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016, pp. 309–320.

[112] ——, "Memory-access aware dvfs for network-on-chip in cmps," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1433–1436.

[113] D. Rahmati, S. Masoudi, A. Khonsari, and R. Sabbaghi-Nadooshan, "Accurate performance bounds calculation for dynamic voltage-freq islands in best effort nocs," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 510–513.

[114] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, and A. Louri, "Dynamic voltage and frequency scaling in nocs with supervised and reinforcement learning techniques," *IEEE Transactions on Computers*, vol. 68, no. 3, pp. 375–389, 2018.

[115] M. Clark, A. Kodi, R. Bunescu, and A. Louri, "Lead: Learning-enabled energy-aware dynamic voltage/frequency scaling in nocs," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[116] Z. Lu and Y. Yao, "Thread voting dvfs for manycore nocs," *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1506–1524, 2018.

[117] Y. Yao and Z. Lu, "Pursuing extreme power efficiency with ppcc guided noc dvfs," *IEEE Transactions on Computers*, 2019.

[118] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for CMPs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 520–533, 2011.

[119] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 378–389.

[120] J. Zhan, Y. Xie, and G. Sun, "NoC-sprinting: Interconnect for fine-grained sprinting in the dark silicon era," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*. IEEE, 2014, pp. 1–6.

[121] A. Samih, R. Wang, C. Maciocco, and T.-Y. C. Tai, "Router parking in power-efficient interconnect architectures," Apr. 11 2017, uS Patent 9,619,006.

[122] R. E. Tarjan, "Algorithm design," *Communications of the ACM*, vol. 30, no. 3, pp. 204–212, 1987.

[123] L. Chen, L. Zhao, R. Wang, and T. M. Pinkston, "Mp3: Minimizing performance penalty for power-gating of clos network-on-chip," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2014, pp. 296–307.

[124] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Ultra fine-grained run-time power gating of on-chip routers for cmps," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, 2010, pp. 61–68.

[125] J. Yin, P. Zhou, S. S. Sapatnekar, and A. Zhai, "Energy-efficient time-division multiplexed hybrid-switched noc for heterogeneous multicore systems," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 293–303.

[126] A. Mirhosseini, M. Sadrosadati, A. Fakhrzadehgan, M. Modarressi, and H. Sarbazi-Azad, "An energy-efficient virtual channel power-gating mechanism for on-chip networks," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1527–1532.

[127] F. Wang, X. Tang, and Z. Xing, "Applying partial power-gating to direction-sliced network-on-chip," *Journal of Electrical and Computer Engineering*, vol. 2015, 2015.

[128] S.-Q. Lian, Y. Wang, and Y.-H. Han, "Dimrouter: A multi-mode router architecture for higher energy-proportionality of on-chip networks," *Journal of Computer Science and Technology*, vol. 33, no. 5, pp. 984–997, 2018.

[129] P. Wang, S. Niknam, S. Ma, Z. Wang, and T. Stefanov, "Evc-based power gating approach to achieve low-power and high performance noc," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 116–123.

[130] H. Zheng and A. Louri, "Ez-pass: An energy & performance-efficient power-gating router architecture for scalable nocs," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 88–91, 2017.

[131] P. Wang, S. Niknam, Z. Wang, and T. Stefanov, "A novel approach to reduce packet latency increase caused by power gating in network-on-chip," in *Proceedings of the Eleventh IEEE/ACM International Symposium on Networks-on-Chip*, 2017, pp. 1–8.

[132] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Run-Time Power Gating of On-Chip Routers Using Look-Ahead Routing," in *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*. IEEE Computer Society Press, 2008, pp. 55–60.

[133] C. H. Kim, J.-J. Kim, S. Mukhopadhyay, and K. Roy, "A forward body-biased low-leakage sram cache: device and architecture considerations," in *Proceedings of the 2003 international symposium on Low power electronics and design*, 2003, pp. 6–9.

[134] S. Niknam, A. Asad, M. Fathy, and A.-M. Rahmani, "Energy efficient 3d hybrid processor-memory architecture for the dark silicon age," in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2015, pp. 1–8.

[135] J. Wang, X. Dong, and Y. Xie, "OAP: An Obstruction-Aware Cache Management Policy for STT-RAM Last-Level Caches," in *Proceedings of the*

Conference on Design, Automation and Test in Europe. EDA Consortium, 2013, pp. 847–852.

[136] J. Ahn, S. Yoo, and K. Choi, "DASCA: Dead write prediction Assisted STT-RAM Cache Architecture," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2014, pp. 25–36.

[137] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 320–331, 2013.

[138] Y. J. Yoon, N. Concer, M. Petracca, and L. P. Carloni, "Virtual channels and multiple physical networks: Two alternatives to improve noc performance," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 32, no. 12, pp. 1906–1919, 2013.

[139] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "Supernet: multimode interconnect architecture for manycore chips," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[140] S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor, "Late breaking results: Improving static power efficiency via placement of network demultiplexer over control plane of router in multi-nocs," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–2.

[141] S. Yadav, V. Laxmi, and M. S. Gaur, "A power efficient dual link mesh noc architecture to support nonuniform traffic arbitration at routing logic," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*. IEEE, 2016, pp. 69–74.

[142] F. Gilabert, M. E. Gómez, S. Medardoni, and D. Bertozzi, "Improved utilization of noc channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, 2010, pp. 165–172.

[143] J. Wu, D. Dong, and L. Wang, "Hm-mesh: Energy efficient hybrid multiple network-on-chip," in *2016 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE, 2016, pp. 404–407.

[144] A. Flores, J. L. Aragon, and M. E. Acacio, "Heterogeneous interconnects for energy-efficient message management in cmps," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 16–28, 2009.

[145] A. K. Abousamra, R. G. Melhem, and A. K. Jones, "Deja vu switching for multiplane nocs," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*. IEEE, 2012, pp. 11–18.

[146] A. Ejaz and A. Jantsch, "Costs and benefits of flexibility in spatial division circuit switched networks-on-chip," in *Proceedings of the Sixth International Workshop on Network on Chip Architectures*, 2013, pp. 41–46.

[147] D. Kline, H. Xu, R. Melhem, and A. K. Jones, "Domain-wall memory buffer for low-energy nocs," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[148] D. Zoni, J. Flich, and W. Fornaciari, "Cutbuf: Buffer management and router design for traffic mixing in vnet-based nocs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1603–1616, 2015.

[149] G. Kim, J. Kim, and S. Yoo, "Flexibuffer: Reducing leakage power in on-chip network routers," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2011, pp. 936–941.

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

[151] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Transactions on Computer-Aided Design of integrated circuits and systems*, vol. 25, no. 12, pp. 2919–2933, 2006.

[152] G. Michelogiannakis and W. J. Dally, "Router designs for elastic buffer on-chip networks," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009, pp. 1–10.

[153] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Extending the effective throughput of nocs with distributed shared-buffer routers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 548–561, 2011.

[154] L.-W. Wang, "Optimal buffering resources allocation of on-chip networks with finite buffers," in *2011 4th International Conference on Intelligent Networks and Intelligent Systems*. IEEE, 2011, pp. 113–116.

[155] F. Daneshgar and M. Taghipoor, "Increase the efficiency of network on chip using buffer sharing mechanism," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 1, p. 79, 2014.

[156] D. DiTomaso, A. K. Kodi, A. Louri, and R. Bunescu, "Resilient and Power-Efficient Multi-Function Channel Buffers in Network-on-Chip Architectures," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3555–3568, Dec 2015.

[157] I. Seitanidis, A. Psarras, K. Chrysanthou, C. Nicopoulos, and G. Dimitrakopoulos, "Elastistore: Flexible elastic buffering for virtual-channel-based networks on chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 12, pp. 3015–3028, 2015.

[158] S. Liu, Z. Lu, and A. Jantsch, "Highway in tdm nocs," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, 2015, pp. 1–8.

[159] D. Bhattacharya and N. K. Jha, "Analytical modeling of the smart noc," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 4, pp. 242–254, 2017.

[160] T. Garg, S. Wasly, R. Pellizzoni, and N. Kapre, "Hoplitebuf: Network calculus-based design of fpga nocs with provably stall-free fifos," *ACM*

Transactions on Reconfigurable Technology and Systems (TRETS), vol. 13, no. 2, pp. 1–35, 2020.

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

[162] T. Moscibroda and O. Mutlu, "A Case for Bufferless Routing in On-Chip Networks," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 196–207.

[163] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A Low-Complexity Bufferless Deflection Router," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Feb 2011, pp. 144–155.

[164] B. K. Daya, L. Peh, and A. P. Chandrakasan, "Towards High-Performance Bufferless NoCs with SCEPTER," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 62–65, Jan 2016.

[165] X. Xiang and N. Tzeng, "Deflection Containment for Bufferless Network-on-Chips," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 113–122.

[166] C. Feng, Z. Liao, Z. Lu, A. Jantsch, and Z. Zhao, "Performance Analysis of On-chip Bufferless Router with Multi-ejection Ports," in *ASIC (ASICON), 2015 IEEE 11th International Conference on*. IEEE, 2015, pp. 1–4.

[167] Y.-T. Chen, J. Cong, H. Huang, B. Liu, C. Liu, M. Potkonjak, and G. Reinman, "Dynamically Reconfigurable Hybrid Cache: An Energy-Efficient Last-Level Cache Design," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 45–50.

[168] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[169] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A Detailed On-Chip Network Model inside a Full-System Simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on.* IEEE, 2009, pp. 33–42.

[170] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on.* IEEE, 2012, pp. 201–210.

[171] S. Arcaro, S. D. Carlo, M. Indaco, D. Pala, P. Prinetto, and E. I. Vatajelu, "Integration of STT-MRAM model into CACTI simulator," in *Design Test Symposium (IDT), 2014 9th International*, Dec 2014, pp. 67–72.

[172] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 31, no. 7, pp. 994–1007, 2012.

[173] J. L. Henning, "SPEC CPU2006 Benchmark Descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[174] C. Bienia and K. Li, "PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, vol. 2011, 2009.

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

[176] F. Oboril, F. Hameed, R. Bishnoi, A. Ahari, H. Naeimi, and M. Tahoori, "Normally-off stt-mram cache with zero-byte compression for energy efficient

last-level caches," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 236–241.

[177] 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*, ser. PACT '08. New York, NY, USA: ACM, 2008, pp. 72–81.

[178] A. Kanduri, M.-H. Haghbayan, A.-M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen, "Dark silicon aware runtime mapping for many-core systems: A patterning approach," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*. IEEE, 2015, pp. 573–580.

[179] F. Oboril, F. Hameed, R. Bishnoi, A. Ahari, H. Naeimi, and M. Tahoori, "Normally-off stt-mram cache with zero-byte compression for energy efficient last-level caches," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 236–241.

[180] P. Zhou, B. Zhao, Y. Du, Y. Xu, Y. Zhang, J. Yang, and L. Zhao, "Frequent value compression in packet-based noc architectures," in *2009 Asia and South Pacific Design Automation Conference*. IEEE, 2009, pp. 13–18.

[181] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "Approx-noc: A data approximation framework for network-on-chip architectures," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2. ACM, 2017, pp. 666–677.

[182] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, "Performance and power optimization through data compression in network-on-chip architectures," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. IEEE, 2008, pp. 215–225.

[183] Y. Wang, Y. Han, J. Zhou, H. Li, and X. Li, "Disco: A low overhead in-network data compressor for energy-efficient chip multi-processors," in *2016*

53nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2016, pp. 1–6.

[184] J. S. Kim, J. B. Hong, J. Y. Kang, and T. H. Han, "Lifetime improvement method using threshold-based partial data compression in noc," in *2018 International SoC Design Conference (ISOCC)*. IEEE, 2018, pp. 269–270.

[185] J. Zhan, M. Poremba, Y. Xu, and Y. Xie, "Noδ: Leveraging delta compression for end-to-end memory access in noc based multicores," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2014, pp. 586–591.

[186] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 243–254.

[187] J. Wang, X. Dong, Y. Xie, and N. P. Jouppi, "i 2 WAP: Improving Non-Volatile Cache Lifetime by Reducing Inter-and Intra-set Write Variations," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 234–245.

[188] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," *SIGMETRICS Performance Evaluation Review*, vol. 19, pp. 5–11, 1991.

[189] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *ACM SIGARCH Computer Architure News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[190] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, Nov. 2005.

[191] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The M5 Simulator: Modeling Networked Systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul. 2006.

[192] "Standard Performance Evaluation Corporation. SPECweb99 design document." [Online]. Available: http://www.spec.org/web99/docs/whitepaper.html

[193] "Hewlett-Packard Company. Netperf: A network performance benchmark." [Online]. Available: http://www.netperf.org

[194] N. Agarwal, T. Krishna, L.-S. Peh, and N. Jha, "GARNET: A Detailed on-chip network model inside a full-system simulator," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2009, pp. 33–42.

[195] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.* IEEE, 2002, pp. 294–305.

[196] M. Vachharajani, N. Vachharajani, D. A. Penry, J. A. Blome, and D. I. August, "Microarchitectural exploration with liberty," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.* IEEE, 2002, pp. 271–282.

[197] X. Zhu and S. Malik, "A hierarchical modeling framework for on-chip communication architectures [soc]," in *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002.* IEEE, 2002, pp. 663–670.

[198] S. J. Wilton and N. P. Jouppi, "Cacti: An enhanced cache access and cycle time model," *IEEE Journal of solid-state circuits*, vol. 31, no. 5, pp. 677–688, 1996.

[199] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 2, pp. 83–94, 2000.

[200] A. Khakifirooz and D. A. Antoniadis, "Mosfet performance scaling—part ii: Future directions," *IEEE Transactions on Electron Devices*, vol. 55, no. 6, pp. 1401–1408, 2008.

[201] A. Khakifirooz, O. M. Nayfeh, and D. Antoniadis, "A simple semiempirical short-channel mosfet current–voltage model continuous across all regions of operation and employing only physical parameters," *IEEE Transactions on Electron Devices*, vol. 56, no. 8, pp. 1674–1680, 2009.

[202] L. Wei, F. Boeuf, T. Skotnicki, and H.-S. P. Wong, "Parasitic capacitances: Analytical models and impact on circuit-level performance," *IEEE transactions on electron devices*, vol. 58, no. 5, pp. 1361–1370, 2011.

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

[204] G. Kurian, C. Sun, C.-H. O. Chen, J. E. Miller, J. Michel, L. Wei, D. A. Antoniadis, L.-S. Peh, L. Kimerling, V. Stojanovic *et al.*, "Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. IEEE, 2012, pp. 1117–1130.

[205] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2007, pp. 3–14.

[206] "International Technology Roadmap for Semiconductors - ITRS 2.0," http://www.itrs2.net/, [Online].

[207] "International Technology Roadmap for Semiconductors. (2010). Process Integration, Devices, and Structures Update," http://www.itrs.net/, [Online].

[208] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proceedings of the INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE*, 1995, pp. 24–36.

[209] M. Müller, D. Charypar, and M. Gross, "Particle-based Fluid Simulation for Interactive Applications," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2003, pp. 154–159.

[210] G. Grahne and J. Zhu, "Fast Algorithms for Frequent Itemset Mining using FP-Trees," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 10, pp. 1347–1362, 2005.

[211] D. Heath, R. Jarrow, and A. Morton, "Bond Pricing and the Term Structure of Interest Rates: A Discrete Time Approximation," *Journal of Financial and Quantitative Analysis*, vol. 25, no. 4, pp. 419–440, 1990.

[212] D. Hristu-Varsakelis and W. S. Levine, *Handbook of Networked and Embedded Control Systems*. Springer Science and Business Media, 2007.

[213] V. A. Pedroni, *Circuit Design with VHDL*. MIT Press, 2004.

[214] A. Dhir, *The Digital Consumer Technology Handbook: A Comprehensive Guide to Devices, Standards, Future Directions, and Programmable Logic Solutions*. Elsevier, 2004.

[215] "XILINX DESIGN TOOLS - ISE DESIGN SUITE." [Online]. Available: https://www.xilinx.com/publications/matrix/Software_matrix.pdf

[216] "ISE Design Suite 14: Release Notes, Installation, and Licensing." [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/irn.pdf

[217] U. Y. Ogras and R. Marculescu, "Analytical router modeling for networks-on-chip performance analysis," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.

Publications Related to Thesis

Journal(s):

- **Khushboo Rani**, and Hemangee K. Kapoor, "Write-variation aware alternatives to replace SRAM buffers with non-volatile buffers in on-chip interconnects." *IET Computers & Digital Techniques* 13, no. 6 (2019): 481-492.

- **Khushboo Rani**, and Hemangee K. Kapoor, "Write Variation Aware Buffer Assignment for Improved Lifetime of Non-Volatile Buffers in On-Chip Interconnects." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27*, no. 9 (2019): 2191-2204.

- **Khushboo Rani**, and Hemangee K. Kapoor, "Investigating Frequency Scaling, Non-Volatile, and Hybrid Memory Technologies for On-Chip Routers to Support the Era of Dark Silicon". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, doi: 10.1109/TCAD.2020.3007555.

Conference Proceedings:

- **Khushboo Rani**, Sukarn Agarwal, and Hemangee K. Kapoor. "Non-blocking gated buffers for energy efficient on-chip interconnects in the era of dark silicon." In 2018 *8^{th} International Symposium on Embedded Computing and System Design (ISED)*, pp. 74-79. IEEE, 2018. (**Best paper award**)

- **Khushboo Rani** and Hemangee K. Kapoor. "Write variation aware non-volatile buffers for on-chip interconnects." In 2019 *32^{nd} International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, pp. 7-12. IEEE, 2019. (**Best paper award candidate**)

- **Khushboo Rani** and Hemangee K. Kapoor. "ZENCO: Zero-bytes based ENCOding for Non-Volatile Buffers in On-Chip Interconnects." In 2020 *57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1-6. IEEE, 2020

- **Khushboo Rani** , Sukarn Agarwal and Hemangee K. Kapoor. "DidaSel: Dirty data based Selection of VC for effective utilization of NVM Buffers in On-Chip Interconnects". In Proceedings of the *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 151-156. 2020.

Other Publications of the Author

Conference:

- A. Kulkarni, C. Joshi, **K. Rani**, S. Agarwal, S. P. Mahajan and H. K. Kapoor, "Towards Analysing the Effect of Snoozy Caches on the Temperature of Tiled Chip Multi-Processors"-*8th International Symposium on Embedded computing and system Design (ISED '18)*, 2018, pp. 230-235, Kochi, India.

- A. Kulkarni, **K. Rani**, S. Agarwal, S. P. Mahajan and H. K. Kapoor, "Towards Analysing the Effect of Hybrid Caches on the Temperature of Tiled Chip Multi-Processors"-*4th International Symposium on Smart Electronic Systems (iSES '18)*, 2018, pp. 52-57, Hyderabad, India.