DIRECTORY-BASED WIRED-WIRELESS NETWORK-ON-CHIP ARCHITECTURES TO IMPROVE PERFORMANCE

A Dissertation by

Kishore Konda Chidella

Master of Technology, Jawaharlal Nehru Technological University, 2010

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment, of
the requirements for the degree of
Doctor of Philosophy

December 2018

DIRECTORY-BASED WIRED-WIRELESS NETWORK-ON-CHIP ARCHITECTURES TO IMPROVE PERFORMANCE

The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Philosophy, with a major in Electrical Engineering and Computer Science.

_____
Abu Asaduzzaman, Committee Chair

_____
Ramazan Asmatulu, Committee Member

_____
Hongsheng He, Committee Member

_____
Ward T. Jewell, Committee Member

_____
M. Edwin Sawan, Committee Member

Accepted for the College of Engineering

_____
Steven Skinner, Interim Dean

Accepted for the Graduate School

_____
Dennis Livesay, Dean

DEDICATION

‘To my parents, friends, and family members’

ACKNOWLEDGEMENTS

I am very thankful to my dissertation advisor Dr. Abu Asaduzzaman for his continuous encouragement and support throughout my research work. His timely supervision on my progress and guidance helped me to complete this research on time. I appreciate his contributions, unconditional support, and funding to make my Ph.D. research productive and thought-provoking. I am always thankful for his constant support and I am glad to be supervised from such an industrious leader. He always had time and patience to guide me accordingly. It has been an honor for me to work for him as a graduate research assistant and graduate teaching assistant. I am also thankful to Dr. M. Edwin Sawan for establishing a Maha "Maggie" Sawan Fellowship for graduate students, and I am very happy to be honored by such prestigious award. I gratefully acknowledge the funding sources from Wichita State University for travel grants, and scholarships.

I express my sincere gratitude and thanks towards Dr. Ramazan Asmatulu, Dr. Hongsheng He, Dr. Ward T. Jewell, and Dr. M. Edwin Sawan for taking time from their busy schedules and serve on my dissertation committee. I take pleasure in recognizing the efforts of all those who encouraged and assisted me both directly and indirectly with my experimental research. I acknowledge the WSU Computer Architecture and Parallel Programming Laboratory (CAPPLab) facilities and research group for providing me with all necessary resources to conduct the research work, prepare the manuscript, and improve the quality of the work and manuscript.

I would like to thank my family, in-laws, for all their love, and encouragement. I thank my parents for the sacrifice they made to see my dreams come true. A special thanks to my brother Ganesh Chidella, who uplifted my intentions, motivated me in hard times and utmost is the financial support.

Most of all for my loving, favorable, patient, and determined wife Sahithi Chidella whose faithful support during all stages of my life and Ph.D. is much appreciated. I am thankful to god for giving me such a nice, kind and understandable wife as my partner. I may not reach this point, without the inspiration of my wife. To my beloved son, Venu Karthikeya and daughter Mishu Ishaanvi, I would like to express my thanks for being good all the time and always making me cherish and reenergized to accomplish the goals.

Kishore Konda Chidella

Wichita State University

December 2018

ABSTRACT

Network-on-Chip (NoC) architectures have emerged as a promising technology for modern computer systems to address the design challenges of high-performance computing systems. Wireless NoC (WNoC) architectures are introduced to improve performance by reducing the core-to-core communication latency. Conventional WNoCs broadcast messages that increase bandwidth-traffic, communication delay, and power consumption. Studies show that directory-based schemes have potential to reduce bandwidth-traffic and improve performance. This work introduces a WNoC architecture with centralized directory (WNoC-CD) and a WNoC architecture with distributed directories (WNoC-DDs) to enhance faster execution by reducing bandwidth-traffic and communication latency. The impacts of uniform and non-uniform distribution of cores into subnets on performance are also studied.

VisualSim software package is used to model and simulate a traditional mesh and the proposed WNoC-CD and WNoC-DDs architectures by processing different communication scenarios. Experimental results show that the proposed WNoC-DDs reduces communication delay up to 20.54% and 5.40%, respectively, when compared to mesh and WNoC-CD. Similarly, the proposed WNoC-DDs reduces power consumption up to 73.56% and 19.97%, respectively, when compared to mesh and WNoC-CD. In a WNoC-DDs, each subnet works independently and resolves communication issues simultaneously. Experimental results also show that the non-uniform subnets help reduce communication delay up to 11.11% and reduces power consumption up to 14.76% when compared with the uniform subnets. Non-uniform partitioning provides flexibility of allocating tasks to different sized subnets as needed and thus improves the core utilization to a greater extent.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Chapter Page

TABLE OF CONTENTS (continued)

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

2D, 3D	Two-dimensional, Three-dimensional
Addr	Address
CAPPLab	Computer Architecture and Parallel Programming Laboratory
CL1, CL2	Cache Level 1, Cache Level 2
CMP	Chip Multiprocessors
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DASH	Directory Architecture for SHared Memory Multiprocessors
E	East
FDMA	Frequency Division Multiple Access
FLC	First Level Cache
FLOPS	Floating Point Operations per Second
GPU	Graphics Processing Unit
HC	Hop Count
HPC	High Performance Computers
Int	Internal Port
KB	Kilo Bytes
MESI	Modified, Exclusive, Shared, Invalidate
ms	Millisecond
mW	Milliwatt
N1, N2	North 1, North 2
NePA	Network Based Processor Array

LIST OF ABBREVIATIONS (continued)

PE	Processing Element
PFLOP	Petaflop
NI	Network Interface
NoC	Network-on-Chip
PWI	Pure Write Invalidate
PWU	Pure Write Update
RC	Resistor-Capacitor
S1, S2	South 1, South 2
SoC	System-on-Chip
TFLOP	Teraflop
W	West
WNoC	Wireless Network-on-Chip
WNoC-CD	Wireless Network-on-Chip with Centralized Directory
WNoC-DDs	Wireless Network-on-Chip with Distributed Directories

CHAPTER 1

INTRODUCTION

Computing is a critical task of modern technology, where it uses computers to manage and process the information. The revolution of computation led to the developments and improvements in designing low-cost microprocessors. According to Moore's law the number of transistors on the chip doubles about every two years. Till recent times we have been able to push more and more transistors on a single chip, but one day we will reach a limit that a transistor may be one atom length, this will be an absolute limit on the Moore's law [1], [2]. Considering the future challenges with respect to transistor size and its limits, computer scientists are embarking on a fundamental shift in how the transistor density on a single chip is used to increase the performance. The increase in transistor number led to multiple ways of increasing parallelism. Initially, the parallelism is introduced on single-core processors. However, Single-core processors are not good enough for complex applications and they have their own limitations in terms of processing speed and adoptable features. Speedup in single-core architectures can be enhanced by increasing clock speeds however they are certain limitations like switching frequency, heat dissipation, etc. [3]. To increase speed, a modern technology, that is multicore architecture evolved, by addressing the problems of single-core architecture in various aspects. More than one core on a single silicon die is considered as multicore architecture and they ensure better performance with less heat dissipation [4]. However, the growth of multicore architectures is ineffective with the existed programming structures and so the development of parallel programming came to limelight, that allows multiple cores to work independently for different assignments [5], [6]. Multicore architectures are used in analyzing data, scientific research, and to solve complex computational problems. Multicore architecture brings various platforms into one roof, such as parallel

processing algorithms, multiple cluster node networks, and computer architectures for the performance improvement of many applications [7]. Multicore architectures are fast and effective in executing the programs with a trade-off on bandwidth, latency, and power consumption [8]. In multicore architectures, cores are coupled together to work concurrently in parallel for increasing execution speed of complex jobs which need multiple operations to be done at a single instant of time. A large and/or complex job is divided into multiple tasks and the tasks are processed concurrently on many cores; this leads to multicore systems [9], [10]. It is necessary that all cores cooperate efficiently for every single computation. In a multicore system, multitasking is done by assigning different sets of tasks to different cores. More cores are integrated on a single die which presents a need of interconnection between two cores and interconnection among the cores leads to long wiring delays, huge power consumption, and other challenges. To address the interconnection issues, Network-on-Chip (NoC) architectures are proposed and is still widely investigated as a scalable and reliable infrastructure communication [11], [12]. However, with NoCs, the major problems are the connection between nodes, latency, and is suitable for only medium number of cores [13]. Wireless Network-on-Chip (WNoC) architecture is introduced to overcome the problems of multi-hop in NoC [14]. WNoC reduce the number of hops by using wireless links in the path and thus reduces latency and power consumption among cores in a multicore architecture [15], [16]. With the increase of multiple cores on a single die, cache coherence is one of the major challenges particularly when they are associated with shared memory [17]. In the present design of a multicore architecture, there are many techniques to address cache coherence such as directory-based protocols [18].

As we know, a processor is the brain of a computer system and is responsible for computing, making logical decisions, and controlling different activities throughout the process.

A helping hand or additional hardware for a processor or Central Processing Unit (CPU) adds an advantage to the system. Hardware accelerator [19], [20] is an additional hardware to any CPU for improving the performance. In this work, we are introducing Stanford Directory Architecture for SHared Memory (DASH) like hardware in a traditional WNoC architecture. DASH architecture provides high processor performance by maintaining coherence among the caches of all the cores and provides scalability of cores [21]. The proposed architecture is a hybrid combination of the WNoC architecture and the DASH architecture. The major goal of the proposed multicore architecture is to reduce the cache latency among the cores by decreasing the number of hops required to travel from a source core to a destination core using the directory and wireless routers.

Multicore or WNoC architectures are traditionally mesh networks. The physical arrangement of network in mesh has several disadvantages such as congestion due to multicasting, latency, power consumption, and unbalanced workloads. To address mesh network issues, solutions such as partitioning cores into subnets are introduced. In multicore architectures, cores are divided into subnets based on different mechanisms and data is transferred between cores based on network topology [22], [23]. The cores are logically divided into subnets, where each subnet has a center core and the center core is responsible to communicate with-in subnets or out of the subnets. The partition of subnets reduces underutilization of cores, latency, and power consumption. Subnets allow the multicore architecture to work in parallel for various assignments. However, finding a center core is a challenge in many cases. Conventionally, subnets are divided uniformly based on the number of cores for that chip. Uniform subnets imply that every subnet has equal number of cores in that architecture. It will be easy to find a center core for a 3x3 core subnet, which has 9 cores. However, it will be a challenging task to find a center core for a 4x4 core subnet, which has 16 cores. The performance can be boosted if the subnet has a closer center

core and the distance is uniform to other cores in its subnet. In general, applications may need minimum to maximum number of cores. Allocating uniform subnets may result some cores in less or no utilization for some applications, and some applications may need more cores than the number of cores fixed in a subnet.

Non-uniform subnets are becoming popular for some applications as they offer better performance when compared to uniform subnets [24], [25], [26]. The non-uniform subnet results are promising and convincing to adopt the new techniques into WNoC architectures. Non-uniform subnets may be better if different number of cores are required by different applications.

1.1 Computer Architectures

Computer system architectures are influenced by the trends of hardware and software technologies. The performance improvement of any architecture depends on its capability in terms of clock speeds, switching operation of transistors at logic level, and hardware/software methodologies. According to Moore's law, the number of transistors per square inch on integrated circuits doubles every year. With this high package density of transistors there are several advantages and disadvantages to make it in complete usage. Several computer architectures are introduced as the days go on. In mid-1990s, single-core architectures are introduced. Single-core processors can only start one operation at a time. To enhance the performance of single-core processor, multicore architectures are introduced in 2004-2006 [27].

1.1.1 Single-Core Architectures

Single-core architectures have only one processor to process instructions. The performance of single-core architectures achieved through the increase of clock speed and transistors count. However, there are limitations in increasing the clock speed as they end up with thermal or heat dissipation issues [28]. Increased clock frequency also brings the issues of switching speed of

transistors. The performance of a single-core architecture can be enhanced if the core accesses the data quickly and it can be achieved with the introduction of a dedicated cache. The cache memory stores the frequent data and so the latency can be reduced if it is not accessing the main memory. However, the size of cache memory is small, and so the latency is a major issue in single-core architectures. So, for complex computations or multitask environments, single-core architectures are not satisfactory. Mostly, the processors manufactured before 2005 are single-core and they are cheap now with the evolution of multicore architectures. Figure 1.1 illustrates a simple single-core CPU architecture, which has an arithmetic logic unit (ALU) and is possible to execute only a single instruction at a time.

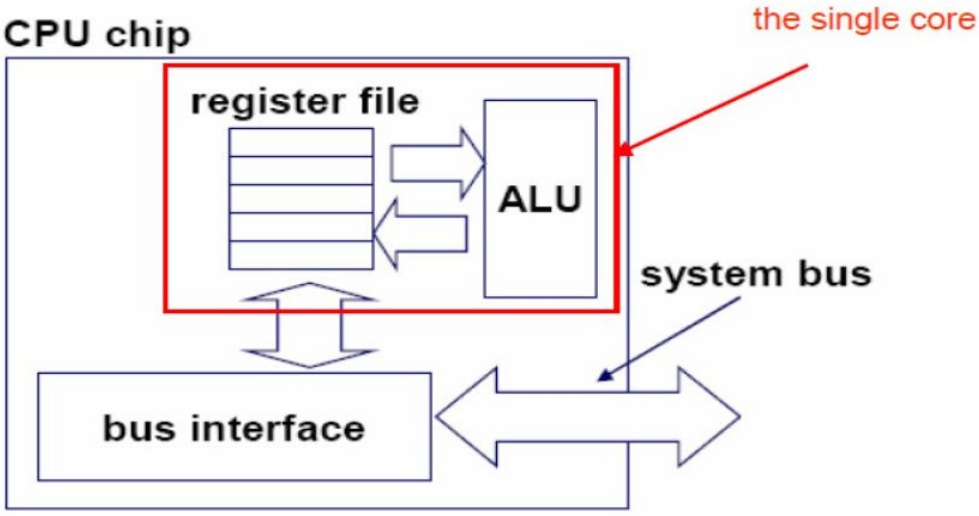


Figure 1.1: Single-core architecture

As time goes on, the modern requirements are not satisfied with the existed single-core architectures as they have certain limitations in multitasking. Single-core based modeling and simulation techniques are not adequate to design modern multicore embedded systems [28]. Multicore processors are emerged to deal with the multiple tasks/applications given to the processor at any given instant of time. Multicore system has multiple cores that can be on a single or multiple system. In multicore architectures, to speed up the computation process, cores are

assigned to perform different tasks in parallel and so they are also termed as parallel processor architectures. Multicore processors execute multiple tasks at the same time, reduces waiting time, and enhances the computer's productivity. However, multicore processors are complex to design and needs more space. Multicore processors are appropriate for more processing power applications and consumes smaller power compared to single-core at the same clock rate.

1.1.2 Multicore Architectures

A multicore processor is a single computing component that has multiple cores and they work independently as a processing unit. Multicore computers are intended to address the issues of single-core architectures like heat and speed. As the name suggests, multicore processing units execute multiple instructions at the same time. The multiple cores are integrated on a single integrated circuit (IC) or multiple die but in a single chip package. Figure 1.2 illustrates the simple multicore architecture of four cores, where each core has its own ALU. The ALU has its own register file and the register files are connected to a shared bus interface.

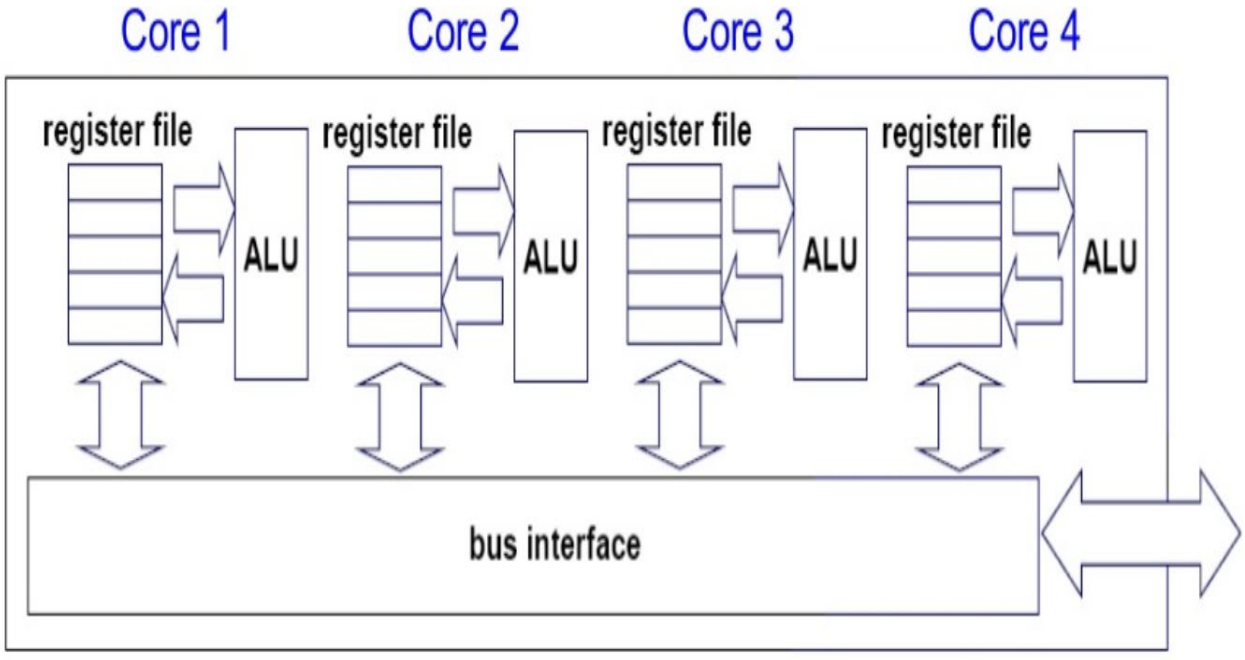
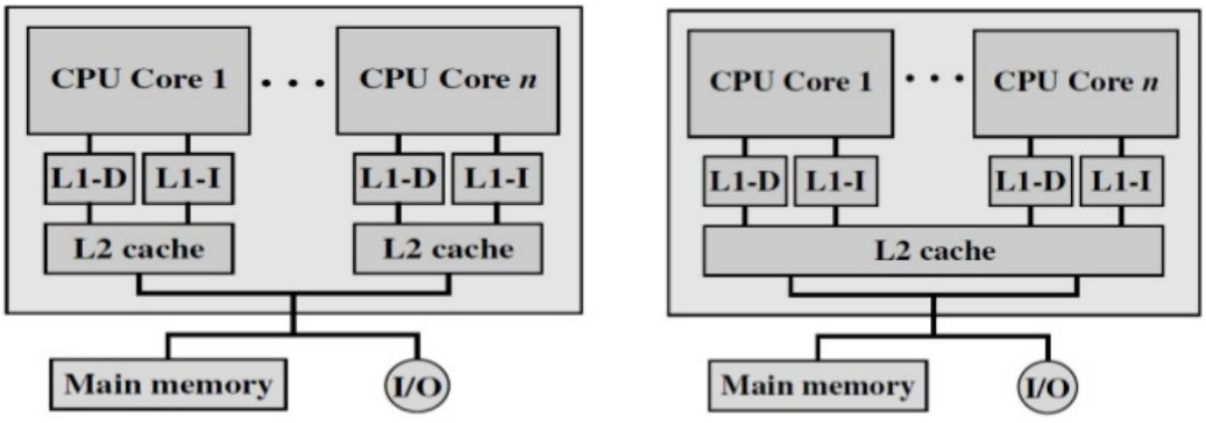


Figure 1.2: Multicore architecture

According to multicore architecture design techniques, four cores running at one fourth of the frequency can approach the performance of a single-core running at full frequency, while the quad-core power consumption is less. If the cores are increased, it would be an advantage for software applications as they have more threads. The multicore architectures are capable to handle multithreaded parallel processing. Multiple threads on multiple cores can be executed simultaneously at the same processor cycle [29], [30].

Multicore systems are designed in a way that two or more cores are coupled together to work concurrently in parallel for increasing execution speed of complex jobs which need multiple operations to be done at a single instant of time. In multicore architectures, speed can be enhanced if the cores access the data quickly and it can accomplish when all the cores have their own dedicated cache. To reduce the latency between the cores, cache levels can be expanded further. However, the performance of multicore also relies on the type of cache utilized such as dedicated and sharing. With the introduction of cache in multicore architectures, cache coherence is a major issue when the cached data from cores is not updated in the shared memory. Figure 1.3 illustrates the multicore architectures organization with dedicated cache and shared cache [31].



(a) Multicore architecture with dedicated L2 cache (b) Multicore architecture with shared L2 cache

Figure 1.3: Examples of cache levels in multicore architectures:

(a) Multicore architecture with dedicated L2 cache (b) Multicore architecture with shared L2 cache

Figure 1.3 (a) illustrates the AMD Opteron organization, where CL1 is divided into L1 data cache and L1 instruction cache with dedicated CL2 for each core. Figure 1.3 (b) illustrates the Intel Core Duo organization that has a dedicated CL1 for instruction and data with a shared CL2 cache. In all the multicore architectures, the main memory is shared and so the cache coherence problems arise if the cached data of the cores are not updated. Multicore architectures are reliable with improved performance for network-on-chip (NoC) architectures. Even though multicore processors have become important, there are still many issues that designers face while designing more than one processing core on a chip. For efficient on-chip communication, there are certain constraints to be considered, such as limited area, communication latency, and power consumption. To combat unnecessary power consumption, many designs incorporate a power control unit which has the authority to shut down unused cores and limits the consumption of power [32]. The bus based multicore architecture [33] is suitable for small number of cores (say, 4-8) with dedicated wires to the cores. However, the manufacturing of chips using dedicated wires would consume more power but offers no or little performance improvement. The inefficiency of dedicated wires resulted in a shift to on-chip networks and incorporating wireless communication among cores. NoC provides a more scalable solution for the multicore architectures.

The scaling difficulties of uniprocessor architectures lead to the evolution of chip multiprocessors (CMPs). To increase the number of cores in a scalable way, the research and evaluation on NoC architectures predominantly increased. The memory hierarchy, interconnect, wiring schemes, routing architecture, network topologies, and power optimization techniques play a key role in the performance of CMP designs as well as NoC architectures. The advanced multicore chip supports several cores say, 10 to 100 or more on a chip and their performance is based on the number of cores and network topology [34], [35].

1.2 Cache Coherence in Multicore Architectures

Single-core architectures are having limitations to speed up by increasing clock frequency as they dissipate enormous heat and consume more power. Then the existence of multicore architectures raised as they are good to distribute work among cores and they can work concurrently to complete the given task successfully. To cut down the costs of multicore architectures, shared memory is introduced. In multicore architecture, cores in a group work together in parallel according to the given assignments and there is a need of data exchange between cores in this process. Due to the cost of memory devices, cache size of each core is limited and so the capacity of it is small when compared to shared memory. In general, the data exchange between cores is through cache and we use different protocols to update the cache accordingly. When the cache is not updated accordingly, then the data in the cache is incoherent and it is generally termed as cache coherence. If the number of cores is less, broadcasting is possible with snoopy based protocols and every core is updated with recent changes on any core. Withal, as the number of cores increases, snoopy techniques are not capable and so the directory-based protocols are developed that gives a room for scalability.

Particularly, the incoherent data may be greater in multicore architectures as they have several cores or processing units. With the number of cores increase, scalability and cache coherence related issues are boosted. To improve the performance of multicore architectures the role of cache is dominant. When a processor/core modifies a cached data element, then it is essential to update or invalidate other cached copies to prevent of usage of obsolete data copies. Cache coherence arises due to non-updated data [36], [37]. However, the performance enhancement majorly lies on cache size, cache level hierarchy such as private and shared. The performance of any system also depends on the type and organization of cache implemented.

Figure 1.4 illustrates the cache organization of a two-core CPU. Here, each core has dedicated cache and the caches share a shared memory resource.

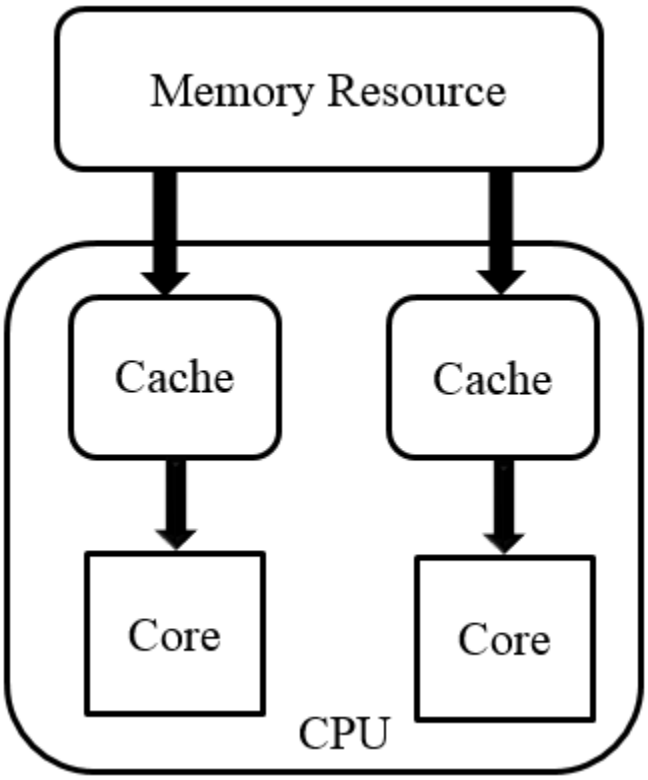


Figure 1.4: Cache organization

Consider a system with many cores, where all of them have their own private cache. The read and write of three of those processors are illustrated in Figure 1.5 (a). After completion of first read at time t_1, processor P_0 will have the value "12" (randomly chosen) in its cache for a variable X which is stored in shared memory location X as illustrated in Figure 1.5 (b). After completion of second read at time t_2, both processor P_0 and P_1 will have the same value "12" in their caches for the variable X as illustrated in Figure 1.5 (c). After time t_3, processor P_0 writes the new value "16" in its cache. In a system without cache coherence mechanism, say it will not be updated to a shared memory location. Therefore, when P_1 and/or P_2 will read next time, they will read the old value "12" as illustrated in Figure 1.5 (d) [38]. To ensure that all the processors

whichever read the new value of X after the update of processor P_0, a new mechanism is required to update the main memory location value as well as all other processors who will be using it.

Time	P_0	P_1	P_2	Memory
t_1	Read X			X=12
t_2		Read X		X=12
t_3	Write 16, X		Read X	X=?

(a) Sequence of reads and writes

P_0	P_1	P_2	Memory
X : 12			X : 12

(b) Cache contents after the read at time t_1

P_0	P_1	P_2	Memory
X : 12	X : 12		X : 12

(c) Cache contents after the read at time t_2

P_0	P_1	P_2	Memory
X : 16	X : 12	X : ?	X : ?

(d) Cache contents after the write and read at time t_3

Figure 1.5: Cache coherence example

1.3 Performance Issues of Network Topologies

Multiple nodes or cores are connected to communicate with each other, which is referred to as network topology. The connection lines between cores are generally termed as hops. The message flow in a network based on the type of topology enforced. The data transfer rate between computers/terminals depends on bandwidth. The higher bandwidth allows the computers to transfer data quickly. There are several network topologies that are available in market with some trade-off among speed, efficiency, and cost. The performance of any network topology depends upon all the components used in that network [39], [40]. To get better performance of a network, using outstanding components for the entire network is essential. The network components are, bandwidth which depends on connection lines, network cards, routers, and cables. On top, the speed and efficiency mostly rely on the type of network, where the computers are connected. A topology can be preferred based on the type of application required as every topology has its own advantages and disadvantages. Out of most, there are some topologies which are extensively used because of their capability with greater trade-off abilities. Namely, some of those trendy topologies are bus, ring, crossbar, and mesh topologies.

Every topology has its own pros and cons. Bus topology is preferred for smaller networks. Bus topology [41] is simple, and they don't have any special computer or controller compared to ring, where it could be useful in controlling the sub network or entire network. The lack of controller in bus topology does not make fit to be adopted in multicore architectures predominantly. In ring topology, computers formed as ring or circular where a neighbor computer is connected to its left and right. Latency is the major concern in this topology, as the number of computers increase in its network, the latency increases accordingly. The major drawback of the ring topology is break in network cabling may affect the entire network. Thus, ring topologies are

not given highest priority in multicore architectures. Crossbar topology [42] is better in multicore architectures when compared to bus and ring topologies. Source node to destination node connections are made through cross bar switch. Each node is connected to all other nodes of the architecture. The major drawback of crossbar topology is the number of switches required and so the cost of the system increases extremely. In contrast to crossbar, the nodes in mesh topology are connected to its own switch. Mesh topologies [43] use blocking technique to get rid of multiple paths especially when there are 3 requests from any specific node. This limitation is mainly due to the routing strategy of mesh, that generally follows XY routing protocol, which indicates only two directions possible at a time for that specific node. The mesh topology follows multiple paths using XY routing protocol to reach the destination [44]. There is a trade-off between the crossbar and mesh topology and if the cost is a major concern, then mesh is the only possible solution.

Wired interconnects can cause delay, power loss, and scalability issues. On-chip interconnects with wireless techniques are introduced to address the issues of wired interconnects. Communication latency and power consumption are important parameters that need to be addressed for improving the performance of architectures with hundred number of cores. There are various challenges to the introduction of wireless routers and directories in WNoC in order to improve the performance of WNoC architectures.

1.4 Problem Description

Network-on-Chip architectures has several issues such as connecting nodes, on-chip temperature, and packaging constraints. Basically, NoC architecture is implemented with a network topology such as bus, crossbar, and mesh. Several challenges encounter based on the type of topology and interconnects used.

In traditional mesh architecture, communication latency, power consumption, and hop count are high due to its architecture design and routing protocol. Mesh architecture is completely wired interconnects and thus having scaling issues. Traditional mesh architectures make use of entire architecture for any application and so they may face underutilization challenges for small applications.

In traditional WNoC, even though wireless routers and clusters division is implemented to address the issues of traditional mesh architecture, it still has the problems with incoherent data, broadcasting, and traffic issues which also increases power consumption. So, a novel architecture is required to reduce wired interconnects, communication latency, cache coherence, data synchronization, and power consumption.

However, as the number of cores increase, the complexity of controlling the architecture in terms of latency, wired/wireless links are always challenging. Instead of using the entire network for a single application, the subnets partition helps to reduce latency and power consumption. The partition of cores into the subnets improve the system performance and they can be categorized into uniform and non-uniform partition. Uniform partition leads to underutilization of cores and more power consumption for smaller applications. Logical partition of subnets to find a center core is always challenging as the number of cores increase mostly if the size of architecture is of even size.

1.5 Contributions

In this work, we propose a novel architecture that enhances performance with minimal energy using wireless routers and directories. Major contributions in this research include:

- Introduction of a centralized directory in WNoC architecture to reduce communication latency and power consumption by addressing cache coherence.

- Introduction of distributed directories to overcome centralized directory issues such as network scalability and performance.
- Introduction of non-uniform partitioning in WNoC to improve core utilization and performance.
- Other contributions include: Introduction of a simulation platform and introduced workload characterization for multicore WNoC simulation.

1.6 Dissertation Organization

The dissertation is organized as follows:

In Chapter 2, literature survey on cache memory hierarchy in multicore architectures, DASH architecture to address cache coherence and data synchronization, popular interconnection network topologies, WNoC topology with wireless routers, uniform and non-uniform partition of subnets in WNoC architectures are discussed.

In Chapter 3, the proposed WNoC architectures with centralized directory are introduced, follwed by distributed directories of 36-core, where each subnet has a single directory. Then, number of cores are extended to 64-core with WNoC-DDs properties. Finally, uniform and non-uniform partitions are discussed.

In Chapter 4, experimental details for this research including assumptions, workload, tools, and parameters are described.

In Chapter 5, some experimental results are presented and discussed to show the performance of various architectures of assorted sizes with different workloads. Finally, this work is concluded in Chapter 6.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we discuss some related published articles as background work and motivation. We start with cache memory hierarchy in single-core and multicore architectures. Then we discuss how DASH architecture addresses cache coherence, and how popular interconnection network topologies such as bus, crossbar and mesh topologies are used. Finally, we discuss WNoC topology and clustering of WNoC cores into uniform and non-uniform subnets.

2.1 Cache Memory Hierarchy

Cache is a hardware that is used to store data close to the CPU to improve performance. Normally, each core has its own cache memory. Single-core architectures can improve performance with increased clock frequency but consumes more power which is nearly 73% with 20% increase of clock frequency. However, with the introduction of a second core, without increasing the frequency, the performance can be improved to 73% with minimal rise of power consumption compared to single-core [45]. Then the designers developed multicore architectures and introduced parallel processing methods such as thread level parallelism (TLP). Multicore supports TLP to boost up performance.

2.1.1 Cache in Single-Core Architectures

To improve the performance of a processor, cache is introduced between the main memory and the CPU. During computation, the core checks its cache for data as/if needed and if the data is not available in cache, it is considered as cache miss. Then the data request is sent to main memory which increases the latency and power consumption. To improve performance further considering single cache issues accommodated with cost and power consumption, cache levels are introduced. The cache close to CPU is cache level-1 (CL1) and then a cache level-2 (CL2) is

introduced to reduce cache miss. CL2 can be shared or dedicated based on the type of architecture. In this research, shared off-chip CL2 is considered. The cache levels are further increased to improve the performance, but they are always shared to cut down the costs. The latency is minimum if CL1 has requested data, but it increases if there is a miss in CL1 and its ascending from there on to main memory. Figure 2.1 illustrates the Celeron processor [46] with 2x16 KB L1 cache and 128 KB L2 on-chip cache levels in single-core architecture. The on-chip CL2 increases the cost of the system.

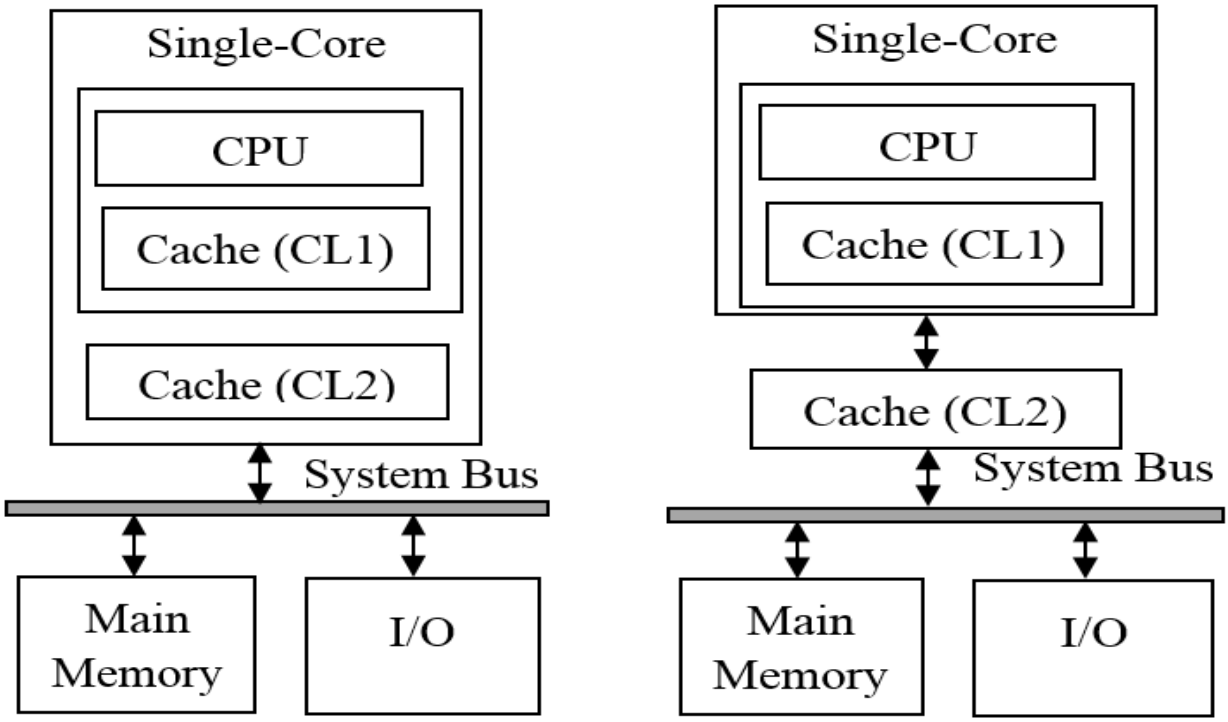


Figure 2.1: Examples of cache organization in single-core architectures: (a) Single-core Celeron processor with private CL1 and on-chip CL2 (b) Single-core Pentium II Xeon processor with private CL1 and off-chip CL2

In some processors, CL2 is off-chip and is close to main memory. Figure 2.2 illustrates the Pentium II Xeon processor [46] with 2x16 KB L1 cache and 512 KB to 2 MB L2 off-chip cache levels in single-core architecture.

2.1.2 Cache in Multicore Architectures

The multicore architecture is a single physical chip that has more than one core. As cores increase, multiple requests to main memory leads to traffic, and latency. So private CL1 is accommodated for each core and thus individual data requests to main memory can be reduced. To incur the costs and improve performance, shared and private CL2 is introduced in multicore architectures [47]. As the number of cores increase, the issues such as cache coherence and scalability rise. Cache follows different techniques or policies to update the data in their system. Cache keeps the data that are frequently referenced, recently referenced, resources near referenced using temporal and spatial locality principles.

Figure 2.2 illustrates the cache hierarchy of Intel quad-core architecture with private CL1 and shared CL2 [48]. The data transfer is faster between cores if the data is obtained from cache and the delay increases if the core misses from individual cache. The lower level caches are always closer to the cores and if there is any data miss in L1 cache, it accesses the L2 cache and it follows further that is main memory if it misses in L2 cache. To work with the cores in great extent, improving cache utilization is one of the workable solutions.

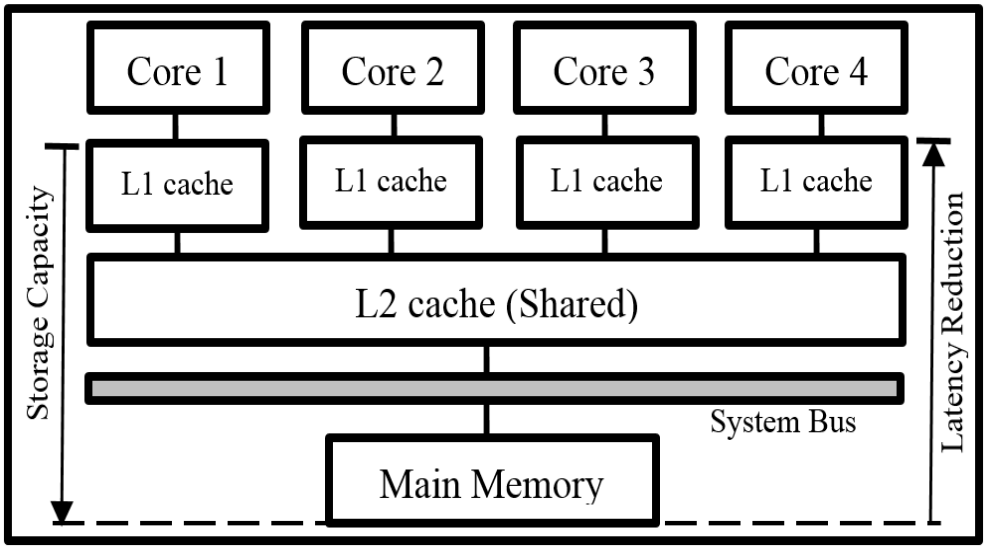


Figure 2.2: Intel-like quad-core architecture with private CL1 and shared CL2

The communication latency to fetch data from cores depends on the level of cache where the data is available. To wind up, the latency and power consumption is maximum when the cores try to fetch data from main memory. To reduce the latency and power consumption, suitable coherence protocols between main memory and cores must be established.

2.1.3 Cache Coherence Protocols in Multicore Architectures

The main reason of using cache is to reduce the execution time of CPUs. If the data is referenced in cache, then it completes the execution in less CPU cycles rather than consuming more cycles when referred to main memory. In multicore architecture, each core has its own cache and so there is a possibility of cache inconsistency and it can be detected dynamically at run-time or statically at compile-time. When multiple copies of data are present in different caches simultaneously, then the problem of cache coherence arises. When the processors can update their own cache freely, then the inconsistent view of memory arises. To ensure data is valid, it is essential to avoid dirty data by using updates or invalidate policies. For any change in data, the processor updates by using write-policy schemes namely write-back and write-through to the cache. In write-back, write operations are made only to cache and the main memory can be valid only if the cache line updates the main memory. In write-through policy, cache as well as main memory are updated at the same time.

There are several cache coherence protocols that are widely used such as snoopy, MESI, and directory-based [49]. The popular protocol to address cache coherence is snoopy protocol [50]. They are majorly used for small core architectures and they follow broadcasting technique. Traditional snoopy methods are popular as they are simple to implement and provides reliable performance at the cost of high bandwidth and a size up to 32 processors only. Broadcasting is a simple technique to find data copies from all other caches, however, it consumes high bus

bandwidth, power, and increase latency for non-shared data compared to shared data. These broadcasting techniques shows that on average, 67% of broadcasts are unnecessary [51]. Traditional pure write update (PWU) protocol has low network latency but high bandwidth required. Traditional pure write invalidates (PWI) protocol has less bandwidth requirement but it has high cache miss ratio. Considering the issues of PWU and PWI protocols, there is a necessity of novel design in protocol that can accommodate for large core architectures.

MESI protocol is one of the popular among all that is basically used to perform write-back to the cache. MESI stands for Modified (M), Exclusive (E), Shared (S), and Invalid (I) [52], [53]. Figure 2.3 illustrates the detailed operation of four states in MESI protocol.

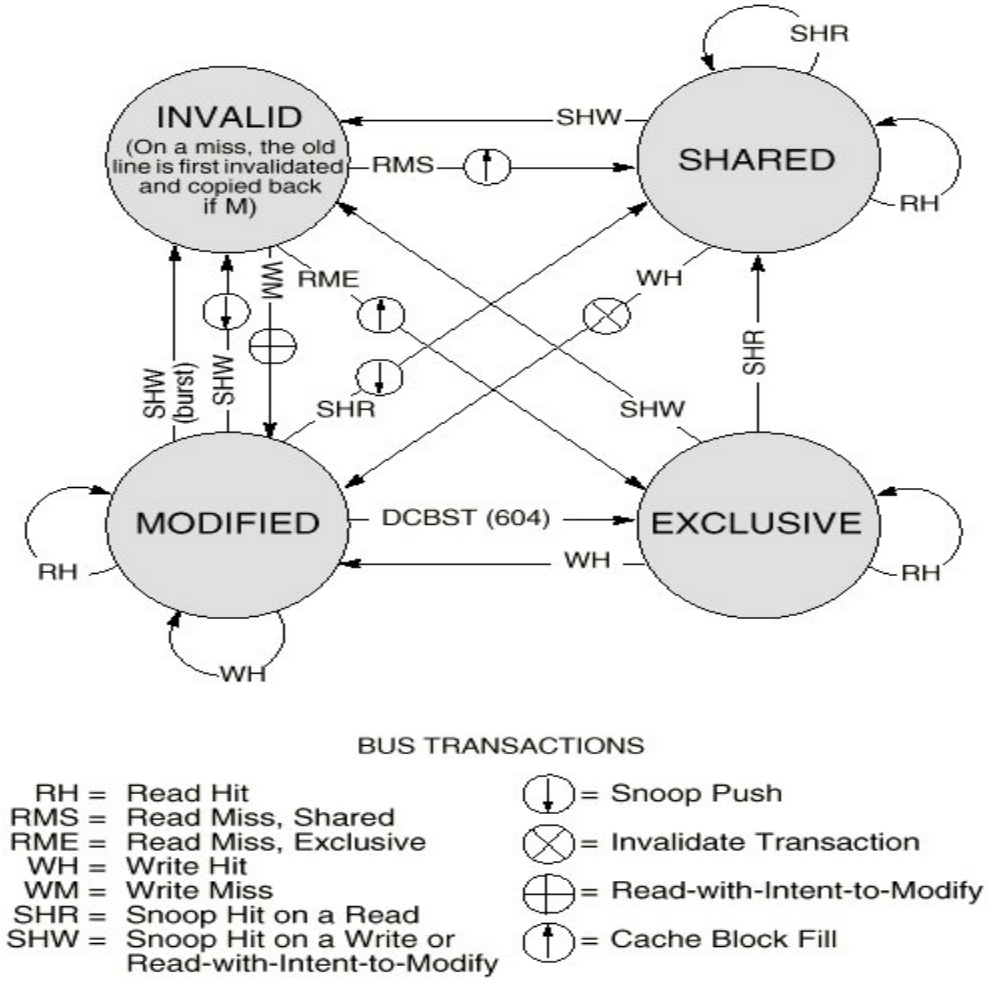


Figure 2.3: Four states of MESI protocol

In MESI protocol, for a read miss, the cache block is moved to either shared or exclusive based on the cache status that is shared or not. If the cache is shared, then the cache will be in shared state, else in exclusive that indicates the data is consistent with main memory. The advantage of MESI protocol is the capability of avoiding bus invalidation. MESI simply skips bus transaction to write to cache instead they move to modified state.

Directory-based cache coherence protocols are better for large core architectures and address the issues of snoopy protocols [54]. Figure 2.4 illustrates the block diagram of directory-based cache coherence protocol. From the Figure 2.4, multiple sharer groups are connected to a shared directory along with L2 cache. Each group individually has different number of processors less than 32 in number and follows a snoopy protocol. Here, the directory receives the requests from each core individually from a sharer group to reduce the network bandwidth. The directory maintains the processor information/data and thus it reduces the latency.

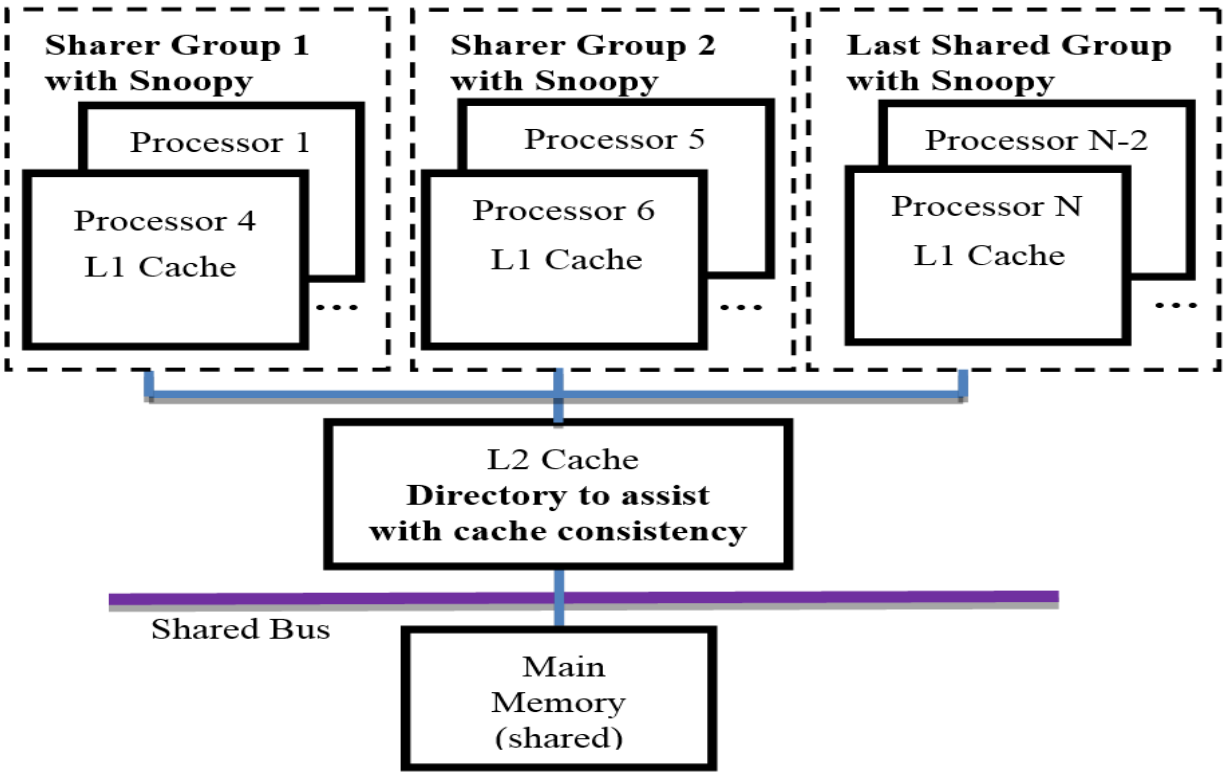


Figure 2.4: Block diagram of directory-based cache coherence protocol

The performance improvement by reducing cache coherence in multicore architectures can be possible with the implementation of directory in cache level (CL2) between main memory and CL1. A directory-based hybrid approach of PWU and PWI reduces cache coherence and the results are promising in reducing bandwidth, memory latency, and cache miss ratio [55].

As discussed above, increased number of cores with increased clock speed may lead to unsustainable power consumption. Alternatively, to enhance performance, parallel programming could be an efficient choice if the instructions dependency is less. The performance of program execution can be improved if the data is retrieved faster from the memory. It depends on the type of cache memory organization used. Distributed memory models offer message passing with improved performance and scalability, but they are complex to design and program. The processor performance can be improved with private and multilevel cache [56]. However, the presence of cache in multicore architectures introduce the cache coherence problem. Hardware and/or software solutions can be used to address cache coherence problems [57].

With the introduction of directory-based architecture, cache coherence problems can be resolved, and data synchronization can be improved. The most popular directory-based architecture is DASH that is scalable, and it is possible to build large scale shared memory architectures. DASH architectures are good to work on parallel applications and so it is a valuable approach to introduce it into multicore architectures.

2.2 Directory-Based DASH Architecture

In this work, we considered Stanford DASH architecture because of its directory-based cache coherence protocol and high scalability. The DASH system supports shared memory architecture inside a cluster of a small number of cores and the message passing technique among the clusters. This architecture provides excellent performance by updating the caches of all the

cores and provides scalability of cores as it does not have any single control unit. DASH protocol does not rely on broadcast messages and instead uses point-to-point messages sent between processors and memories to keep caches consistent [58]. Figure 2.5 illustrates the high-level organization of a DASH system [21].

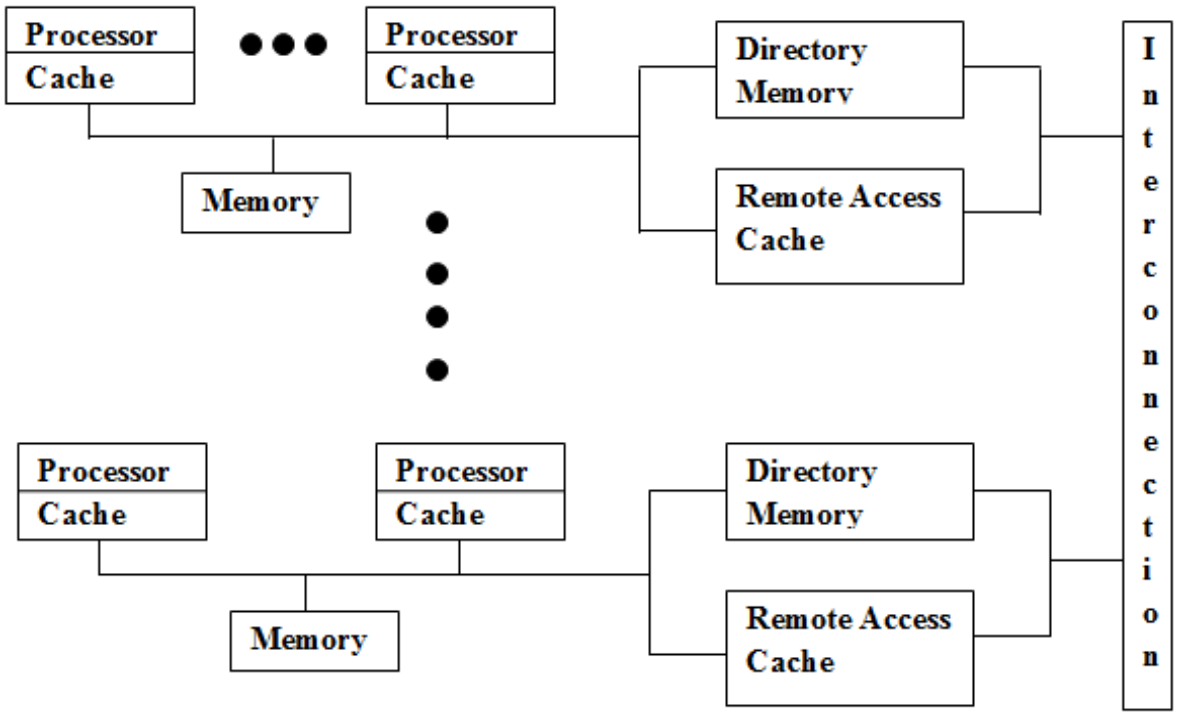


Figure 2.5: DASH architecture for shared memory

Typically, a DASH system may consist of many processing nodes via an interconnection network which has large bandwidth and a low communication latency. The physical memory or the main memory is distributed among all the clusters in such a manner that the memory is accessible for every core. Each processing core has its own individual cache. To maintain cache consistency among the cores of a cluster a bus-based snoopy scheme is used, and a distributed directory-based coherence protocol is used to maintain cache consistency among the clusters. In DASH architecture, shared memory provides a major reduction in the communication latency.

2.3 Interconnection Network Topologies

In this subsection, we discuss some popular network topologies such as bus, crossbar and mesh. In parallel architectures, network topologies refer to the type of interconnections technique among multiple cores and memory modules. Every network topology has its own pros and cons. There will be always a trade-off between speed, hop count, and power consumption based on the chosen topology. If one or more devices connected to each other for inter-device communication, then the system is considered as interconnection network. Interconnection networks are especially used to connect processors/cores to processors/cores. Typically, the cores might be connected to private level cache memory and shared memory. Depending upon the type of interconnections and memory module entities, the performance of parallel or multicore architectures is derived. Low latency and cache coherence problems are challenging in NoC [59]. Generally, when a processor with memories are connected to each other, then we call it as node. Based on the node's interconnection, performance parameters such as scalability, reliability, applicability, and cost can be supervised while designing an efficient multicore architecture.

2.3.1 Bus Topology

In bus topology, all nodes are connected to a main cable that has terminators at both ends [60]. When a node sends data signal, it will flow in both directions of cable. At each end of the cable, the signal absorbed by terminators to avoid signal bouncing. Signal bouncing should be avoided to overcome the chances of collision, when two or more nodes are trying to send the signal at the same time. The nodes in bus network topology are connected in a linear method and is illustrated in Figure 2.6. Bus topology is simple and cheap to implement. The topology requires less cable compared to star topology and it is most appropriate in smaller networks. The

terminators wouldn't be expensive, and the network doesn't require any additional hubs or switches to establish communication between nodes.

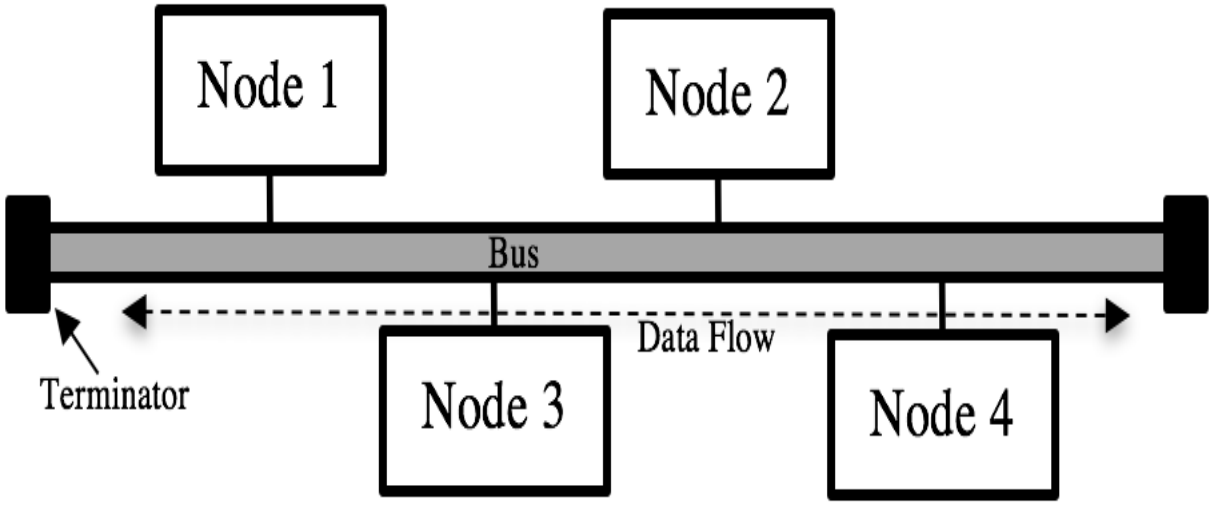


Figure 2.6: Bus network topology

In bus topology, if single node is down, then it wouldn't affect the entire network. However, if the bus or main cable fails then it affects the entire network. Additional devices can be easily connected to the network. But the performance can be degraded with increased nodes, data size, and not suitable for heavy traffic [61]. The central cable length has a limit and thus the number of nodes connected to cable, which brings the issues of scalability. In case of time-shared common bus, only a single communication between two processors or access of main memory is possible with a limited transfer rate. Also, the troubleshooting is difficult to manage in large networks.

2.3.2 Crossbar Topology

In crossbar topology, the switches are arranged in a matrix configuration that has multiple input and output lines as illustrated in Figure 2.7. Crossbar switch topology is a low latency and high throughput network [62]. In crossbar topology, every node is connected to other node with non-blocking feature. The arrangement of cores in crossbar topology is in rows and columns

pattern. The crossbar topology achieves high performance as the switches provide all possible permutations [63].

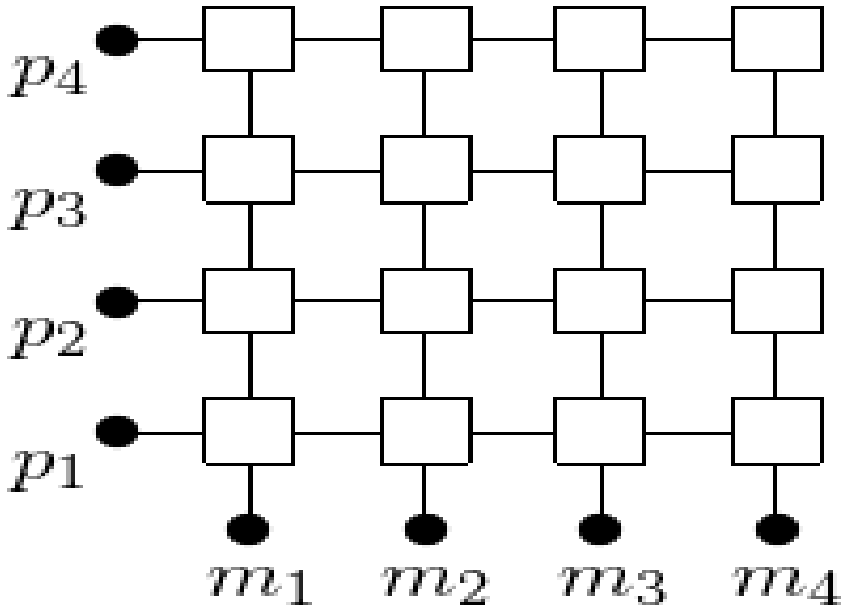


Figure 2.7: Crossbar topology

In cross topology, every node can reach other node through the corresponding switch by following a XY routing algorithm. The number of horizontal and vertical links are interconnected by a switch and the communication between nodes is through these intersections. In crossbar, to select a node the topology has unique intersection. There is no alternative path if any node in row or column fails.

As illustrated in Figure 2.7, the crossbar network uses *p*m* grid matrix to connect *p* inputs to *m* outputs in a non-blocking manner. The crossbar topology provides higher bandwidth with reduced hop count. Crossbar supports simultaneous transfers from all memory modules and possibility of considering alternative switching route. However, the crossbar topologies have drawbacks such as failure of any cross-point prevents the communication between those intersection points. The cross-points are inefficiently utilized as every node is not extremely

engaged in every communication. Crossbar topologies are expensive as they require many wires and lack of scalability, because the crossbar needs N^2 switches for N nodes.

2.3.3 Mesh Topology

Mesh topology is simple, and it can reach destination through several paths. Mesh is easy to layout on-chip with equal length of links. Mesh is a potential network topology for multicore architectures [64]. In a two-dimensional (2D) mesh network, all cores are connected in a crossbar connection as illustrated in Figure 2.8. The cores are plotted/organized in rows and columns method and they are addressed using matrix technique. Mesh network topology is the most common topology used, due to its advantages of shorter wavelength, low router complexity, and feasibility.

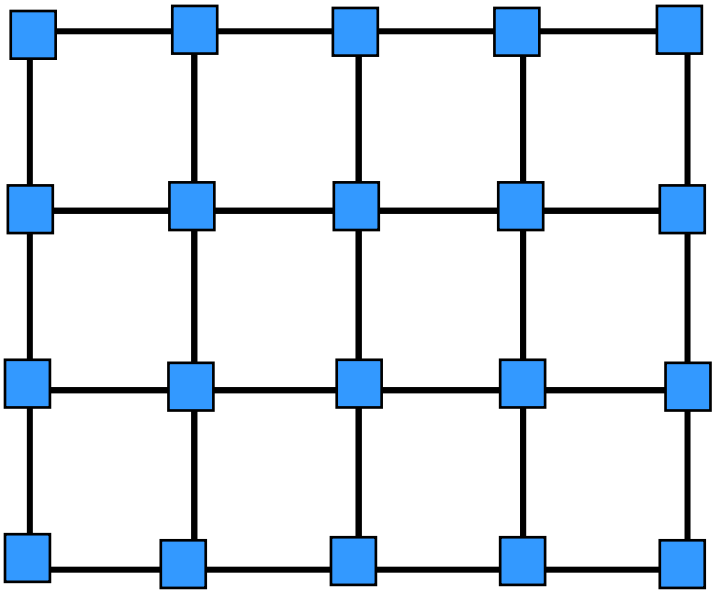


Figure 2.8: 2D Mesh topology

Wired mesh network provides very good reliability for inter-core communication [65]. In realistic implementations, 2D meshes with equal number of nodes along each dimension are used for connecting a set of processing nodes. The mesh topology with XY routing algorithm has several advantages such as never runs into deadlock or live lock.

In mesh topology, the addresses of the routers can be simply determined as XY coordinates in mesh [66]. When the source column is different from the destination column, initially a packet moves through horizontal axis and then it takes vertical axis to reach its destination. There are many routing algorithms used by various topologies to reach the destination core. In NoC architectures, mesh architecture is considered as a root architecture and on top of that extension of new techniques such as wireless routers are added to get advantage of performance such as speed and scalability. However, traditional mesh topology has many disadvantages, such as network congestion, poor scalability, high power consumption, and long latency. Traditional mesh topology in NoC also faces traffic issues and multiple path policies to reach destination. Adequate control unit to address traditional mesh challenges is required to improve the performance further.

2.4 Wired-Wireless Network-on-Chip Topology

Wireless network-on-chip topology basically developed on top of mesh architecture [67]. In mesh architecture, all the cores are accustomed to do a single task that may or may not have subtasks. In this strategy, some of the cores may not be utilized and thus brings underutilization issues and consumes more power as all the cores are active. So, to enhance the utilization and performance, new principle must be introduced that allows small number of cores out of maximum available cores to work in a group for a single task. This method of grouping cores is generally termed as clustering cores into subnets.

2.4.1 Clustering Cores into Subnets

Clustering is introduced to improve the performance of multicore architectures. When several cores of same kind are grouped together as a bunch, then it is called clustering. In Figure 2.9, 36-core mesh architecture is divided into 4 subnets, and each subnet has 9 cores.

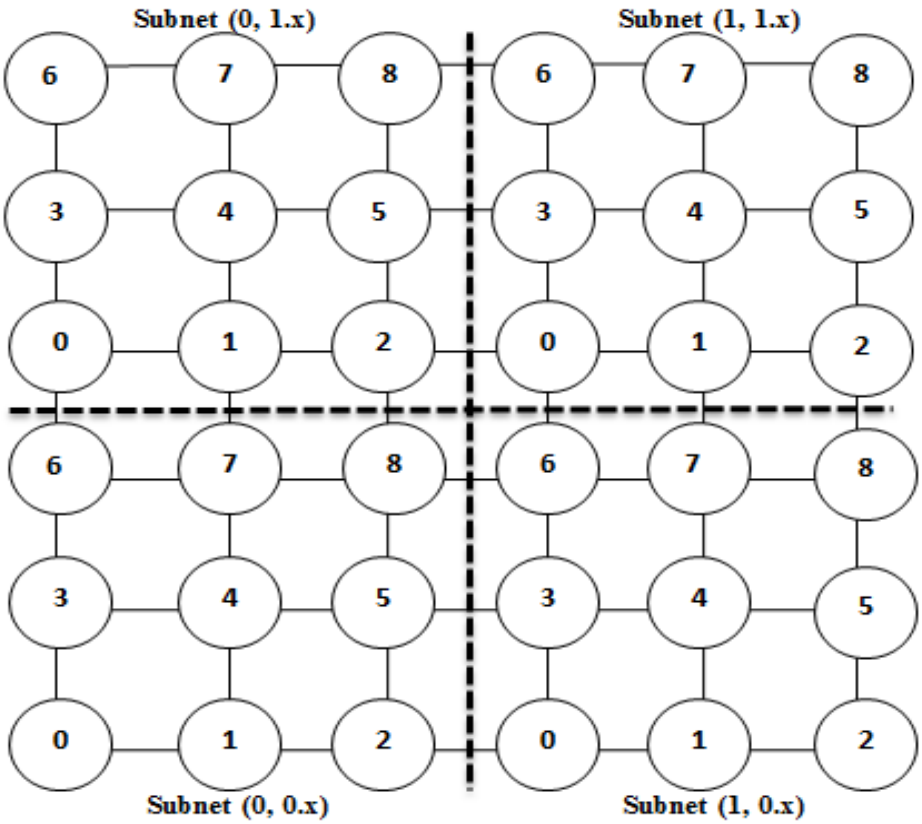


Figure 2.9: Mesh topology with subnet division

Instead of using the entire network for smaller workloads, the cores are divided into clusters which gives the scope of assigning multiple tasks that uses a single cluster or multiple clusters according to the given workload. This virtual clustering allows the network to be active or non-active cluster according to the given task. This will help in reducing the power consumption as idle network consumes less power compared to active cluster. The subnet division will make an individual small network and it could reach the destination faster if the destination is in the same subnet.

Even though, the cores are clustered into subnets, at some point they need to follow traditional mesh topology that has multiple path policy to reach the destination. This method increases latency and power consumption. To address such issues, alternative routing with the wireless routers is introduced.

2.4.2 Wireless Routers into Subnets

To enhance the performance or to reduce latency of traditional mesh clustering, wireless routers are introduced [68], [69]. These routers avoid traditional routing and follows subnet to subnet communication with only one hop which reduces the latency. In other words, wireless routers reduce the number of links between source and destination compared to traditional mesh. Each router is having its own processor and control logic to manage the data sending or receiving to other subnets or its own subnet. WNoC architecture is basically a network-based processor array (NePA) [70]. NePA is a two-dimensional *row* x *column* processor array with mesh topology. The key ingredient in NoC design is based on decoupling of computation from communication. Each processing element (PE) consists of a processor core, network interface (NI), and a router. The processing core takes care of every task and are responsible for data synchronization. The architecture of a NePA is illustrated in Figure 2.10.

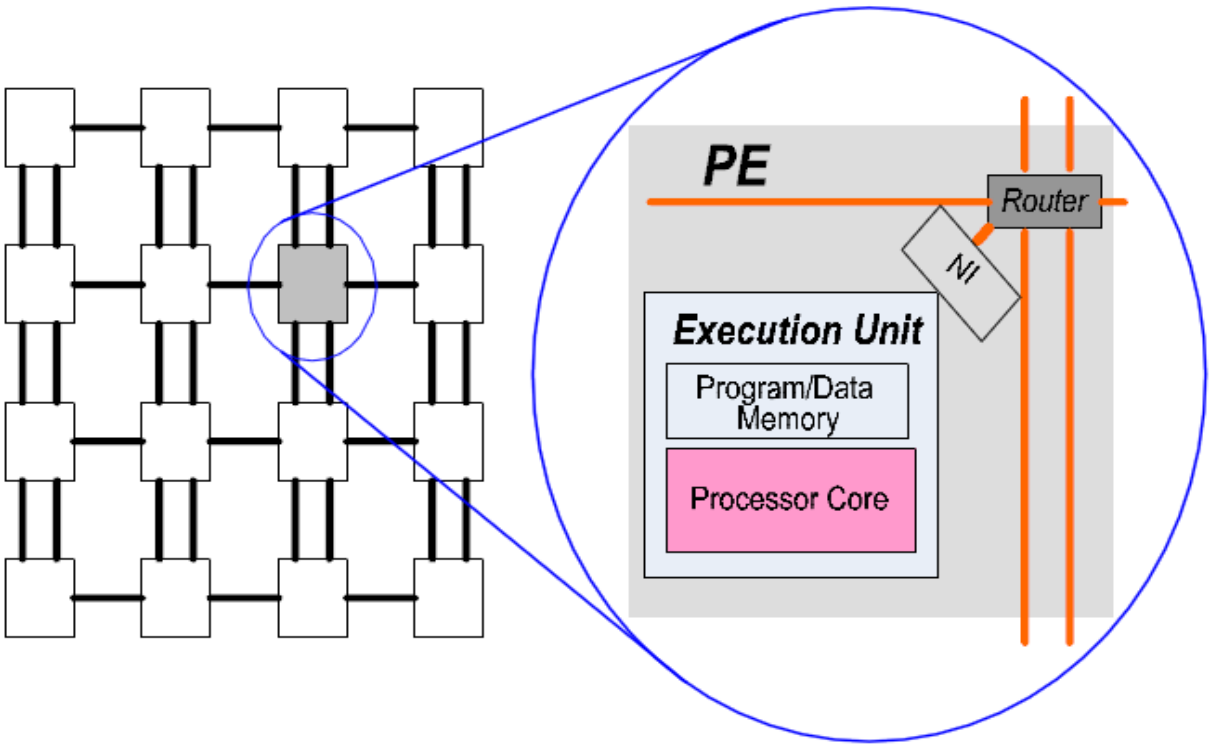


Figure 2.10: 2D NePA architecture with 4X4 matrix

The routers in NePA architecture has two bidirectional 64-bit links connecting it with the neighboring routers and additionally they also have vertical ports. With the help of the links, two subnets can be formed – an East subnet and a West subnet, separating the whole network into two sub-networks. The input and output ports of a NePA router is illustrated in Figure 2.11.

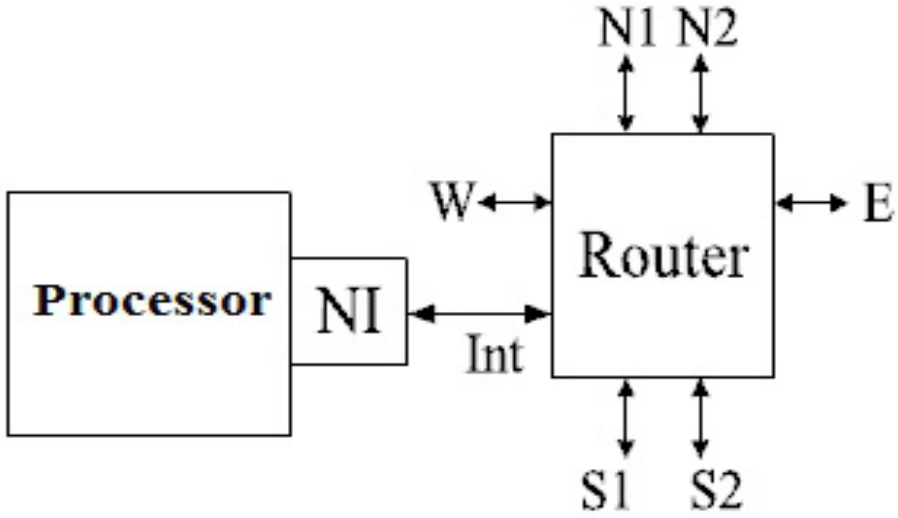


Figure 2.11: Port description of NePA router

Whenever a packet is to be transmitted it is injected into the router via internal port (Int) and accordingly it is directed to destination by directing it towards either East-subnet or West-subnet. NePA utilizes an adaptive XY routing [71] scheme to route the packet from source to destination. To balance the link utilization and improve network performance, the router selects an alternative output port for incoming packets. This process is useful, especially when the output port is congested. Wireless routers are capable of transferring packets via wired as well as wireless.

Some of the wired routers in WNoC are replaced with wireless routers which have wireless links to other routers in different subnets, in addition to the original wired links. Figure 2.12 illustrates the traditional WNoC architecture, where the cores are divided into four rectangular subnets and the wireless routers are placed in the central core of each subnet.

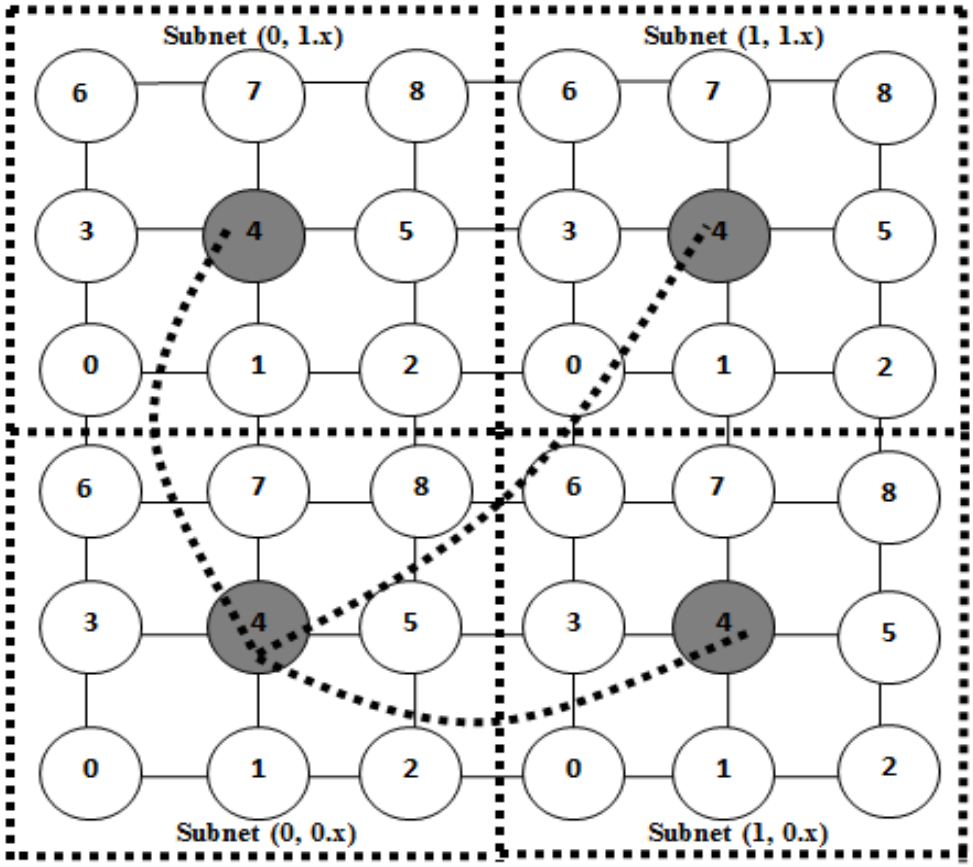


Figure 2.12: Traditional wireless network-on-chip architecture with wireless routers

WNoC is capable of transferring packets through wired and wireless links [72]. In WNoC, processing cores are divided into various subnets, where each subnet has one wireless router and is responsible to broadcast the requested data to all other subnets [73]. In Figure 2.12, the dotted and curved lines represent the wireless links and the solid lines represent wired links among the processing cores to transmit the data packets between routers.

The frequency division multiple access (FDMA) technique is chosen to provide simultaneous communication among the multiple wireless routers. Transmitter and receivers installed on a wireless router are assigned with an independent carrier frequency to accommodate data from different channels. Wormhole packet switching, which offers many advantages such as lower transfer latency and a low buffer requirement, is used to transfer packets of data among the

cores. The whole network is divided into subnets and each node is identified within its subnet using a local address. The features of addressing a specific core in a network help WNoC provide much faster routing decisions as well as a scalable hierarchical system. However, the traditional WNoC architectures has broadcasting and bandwidth issues. These issues can be addressed if a directory is added on top of traditional WNoC architecture.

2.4.3 Uniform and Non-Uniform Partition of Subnets

When several applications are running in a multicore NoC architecture, the amount of traffic generated is significant. The traffic generation is based on multiple loads/applications to the network. The performance of NoC architectures can be critical if the traffic is extensive. In multicore architectures, the traffic relies on workload as well as subnet mechanism. As we know, clustering cores into subnets reduces hops and thus reduces latency as well as power consumption [74]. Therefore, the method of clustering plays a key role in improving the performance. For example, if the application is using the cores in only one subnet, then it is not essential to request the data from other subnets. This will reduce the waiting time, processing time, and data transfer time. However, if the size of the subnet is too large then it is not possible to run multiple applications on a subnet. At the same time for small applications some cores may be idle and can be classified as underutilization. This will cause increase in latency and power consumption. Hence, there is always a tradeoff in determining the size of subnets. In general, for small and same sized applications, uniform partition of subnets can bring good benefits in improving performance.

To avoid traditional network congestion of mesh topology wireless router is introduced in subnet. Too many wireless routers will also increase traffic and channel interference. Wireless routers allow the subnets to communicate directly rather than core to core, which happens in traditional mesh topology. The subnet to subnet communication reduces the links and traffic. With

this technique, a subnet can run its own application alone and can serve the requests of other subnets with a minimal complexity and delay. The other key factor of determining subnet size and assigning wireless router is based on the number of cores. If the wireless router is in center to a subnet, then the performance of subnet is exceptional as the neighbor cores are approximately equal in distance to the center core. If the size of subnets in any architecture are equivalent, then they are classified as uniform partition of subnets.

Uniform partition of subnets is entertained for smaller core architectures. So more than one subnet may be involved for larger applications. The latency and power consumption rise as the number of subnets involved for any application. As the number of cores increase, determining subnet size and center core are complicated without compromising the performance of NoC architectures.

Non-uniform partition of subnets is recommended to address the complexities of uniform partition [75], [76], [77]. The scope of various size of subnets satisfy distinct largeness of applications with reduced latency by minimizing hops. Uniform partition with even number of cores like 4, 8, 12, 16, etc. have the difficulties in determining approximate center core. The shift of center core varies the performance of the system and it benefits certain cores dominantly, which are nearly connected to center core. Thus, in larger core NoCs, non-uniform partition to avoid even size subnets is preferred. With this approach, determining center core and making availability of a subnet to large applications can bring stability to improve performance.

2.4.4 Adaptive XY Routing Algorithm for Wireless Network-on-Chip Architecture

Adaptive XY routing algorithm is efficient in fully utilizing network resources. Basically, adaptive XY routing is a subsidiary of traditional XY routing algorithm. Adaptive routing depends on the neighbor's load condition to make a route between source node and destination node. If the

congestion is too high, then the nodes check for an alternative route that has less congestion path. Each node has horizontal path with 2-bits quantized value and vertical path with 2-bits quantized value, which totally makes 4-bits load value to find the less congestion path. Horizontal path node uses 2-bits quantized value that reflects East subnet and West subnet to calculate the less congestion path. Similarly, vertical path node uses 2-bits quantized value that reflects North subnet and South subnet to calculate the less congestion path.

To establish a route between source node and destination node, configuration packets are generated with the collaboration of neighbor nodes. Adaptive algorithms may need more computation than deterministic algorithms to identify the correct path for sending packets between nodes [78], [79], [80]. The performance can be improved when the load is uniformly distributed throughout the network and maintains balanced nature of the architecture.

CHAPTER 3

PROPOSED DIRECTORY-BASED WIRED-WIRELESS NETWORK-ON-CHIP ARCHITECTURES

In this chapter, we introduce our proposed directory-based wired-wireless network-on-chip architectures. We describe the design considerations and working principle of the directories. We propose three architectures as listed below:

- Proposed Architecture 1: Introduction of Centralized Directory in WNoC Architecture with Uniform Partition of Subnets
- Proposed Architecture 2: Introduction of Distributed Directories in WNoC Architecture with Uniform Partition of Subnets
- Proposed Architecture 3: Non-Uniform Partition of Subnets in WNoC Architecture with Distributed Directories

The proposed architecture is a hybrid combination of the WNoC architecture and the DASH architecture. The major goal of the proposed multicore architecture is to reduce the communication latency among the cores by decreasing the number of hops required to travel from a source node to a destination node using the directory and wireless routers. The key design considerations include: grouping cores, designing directory, managing cache consistency, and communication among cores.

Primarily, in this work, we introduce a single directory that is centralized directory for 4 subnets, where each subnet has 9-core that makes a total of 36-core architecture. Thus, we design a novel architecture, that is wireless network-on-chip architecture with centralized directory (WNoC-CD). We model all the architectures using VisualSim tool and derive the performance

characteristics such as communication latency, hop count, and power consumption. The proposed centralized directory is compared with traditional mesh and traditional WNoC architectures [81].

However, centralized directory is not suitable for larger networks. The load on centralized directory could be heavy with larger networks and thus drawbacks such as delay, data synchronization, traffic and bandwidth issues may arise. To overcome the issues of centralized directory, distributed directories are introduced in WNoC, that is WNoC-DDs. The performance of WNoC-DDs is compared with traditional mesh, traditional WNoC, and WNoC-CD.

As the number of cores increases, the challenges of enhancing performance increase. The performance of directory introduced to subnets will increase the overall performance. However, selection of center core that hosts the directory plays a key role in performance improvement and it could be better if the center core is in equal distance or closer to the other cores in its subnet. For large core architectures, the size of subnet is large and allocating a full subnet for low loads leads to underutilization of network and boosts power consumption. So, uniform partition of subnets for large core may not be satisfactory. Also, uniform subnets may not be suitable for different-sized applications. Considering the weaknesses of uniform partition, a non-uniform partition approach is examined.

3.1 Designing Directories for WNoC Architectures

In the design of centralized directory or distributed directories in WNoC architecture, the basic abstraction is identical. In both centralized and distributed directories, the purpose of the directories is to hold information about the cached copies. A powerful processor with a wireless router is used to host the directory. The center core of each subnet is integrated with a wireless router. The wireless router is capable of transmitting and receiving the data between the subnets.

The directory contains the information of all other subnets that includes data sync, minimal routing path, and it is integrated with wireless router in the central core of each subnet.

WNoC-CD architecture has a single centralized directory and the directory is responsible for providing information about the cached copies. WNoC-DDs has distributed directories, where all directories are identical. The directory contains cores' subnet addresses, the status of each cached block, and the addresses of the blocks that have been cached. The directory is dynamic in nature and the total number of directory entries depends on the number of cache blocks/lines per core. It is explained below with an example:

Say, the cache size per core is 1 KB (1024 Bytes) and the size of each cache block (also known as, cache line) = 128 Bytes. So, the number of cache blocks = Total size of memory in cache / Size of each cache line = 1024 Bytes /128 Bytes = 8. Therefore, for an n-core system, n x (1 + 8) entries are required. In each row, one column for the core number and eight columns for eight blocks. Table 3.1 illustrates a row in directory that shows the initial stage of Core-1. Initially, the blocks for each core in the directory will be empty. Whenever a core caches data, the selective block address of the specific data is recorded to the corresponding block of that core. Table 3.1 illustrates initial stage of Core-1, Table 3.2 illustrates the changes after reading from Core-1, and Table 3.3 illustrates the changes after write operation to Core-1.

Table 3.1: A row in directory that shows initial stage of core-1

Core #	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Core1 (0,0.0)	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty

Table 3.2: A row in directory showing changes after reading a block by core-1

Core #	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Core1 (0,0.0)	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	E 100th Blk	0 Addr Empty	0 Addr Empty	0 Addr Empty

Table 3.3: A row in directory showing changes for write in a block of core-1

Core #	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Core1 (0,0.0)	0 Addr Empty	0 Addr Empty	0 Addr Empty	0 Addr Empty	M100th Blk	0 Addr Empty	0 Addr Empty	0 Addr Empty

The status of every subnet before and after the requests of data with-in or out of subnets is controlled and monitored by the directory. The cache size of each core is split into cache block/cache line. When the directory receives the request for data from a core, then the directory checks the block individually and is defined as an entry. The directory controls and monitors the status for the data requests with-in or out of the subnets. The number of entries required for a request depends on the cache size and block size. The parameters for a cache size, block size, number of entries is listed in Table 3.4.

Table 3.4: System parameters of a directory

System Parameters	**Relevant Value**
Cache size/core	1 KB
Each cache block size	128 Bytes
Number of cache blocks/core	8
Number of entries/ 9-core	81
Number of entries/ 36-core	324
Number of entries/ 64-core	576

3.2 Customizing MESI Protocol for WNoC Architectures

The working principle of both central and distributed directories is identical. A few additional steps are required for distributed directories to maintain data synchronization between directories of all other subnets. The directory working principle is explained in detail by considering a scenario of Core-1 requesting for data and the required data blocks are cached from the main memory to the cache of Core-1. The information regarding the cached block will be stored in the directory as well. At the beginning, the directory is empty and alterations to the

directory are made according to the outcomes of the requests for blocks made by the cores. For a 1KB cache with 128B lines:

(1) Initially all the blocks of directories would be empty with Status '0' as shown in Table 3.1

(2) After Core-1 makes a request to fetch 100^{th} block of the next level cache/memory, the block number is 100 mod 8 = 4. The fetched data is stored in the 4th block of the cache of Core-1 (see Table 3.2) with Status 'E' (for Exclusive).

(3) If the same/cached data is read again by Core-1, then there will be no change in Table 3.2.

(4) If Core-1 performs a write operation on the cached block, then the status of the block will be changed to 'M' to indicate that the value is modified (as illustrated in Table 3.3). A protocol to manage cache consistency is explained next.

With respect to the read/write requests, the state of a block in the directory changes accordingly as illustrated in Tables 3.1, 3.2, and 3.3. The directory keeps track of each cached block and maintains its state. The directories are updated, if there is any request of read or write to any core and thus it can send the data requested by a core.

Another example: when a core requests for a read operation for the first time, the data of that selective memory location is read and stored in the appropriate block, the directory is updated with an 'E' (for Exclusive) and the block address. When a core requests a write operation on the same block, the state of that selective block is changed to 'M' (for Modified). For every write operation, the directory is updated with an 'I' (for Invalidate) for the other cached copies. The state 'S' (for Shared) of a cached block means more than one cores are sharing that selective block. In case of multiple directories, updating/synchronizing directory is essential to avoid data inconsistency.

3.3 Proposed Architecture 1: Introduction of Centralized Directory in WNoC Architecture with Uniform Partition of Subnets

At the beginning, a DASH architecture with wireless routers is introduced in a traditional WNoC architecture. Each subnet has equal priority to the centralized directory.

3.3.1 Clustering Cores into Uniform Subnets of WNoC Architecture

The proposed architecture divides the cores on the die into clusters called subnets and a directory with wireless router is introduced to improve the performance of traditional mesh and WNoC architectures. In each subnet, one special core, such as core-4 in subnet (0, 0.x) of Figure 3.1, contains a wireless router and the other cores within subnet contains wired routers.

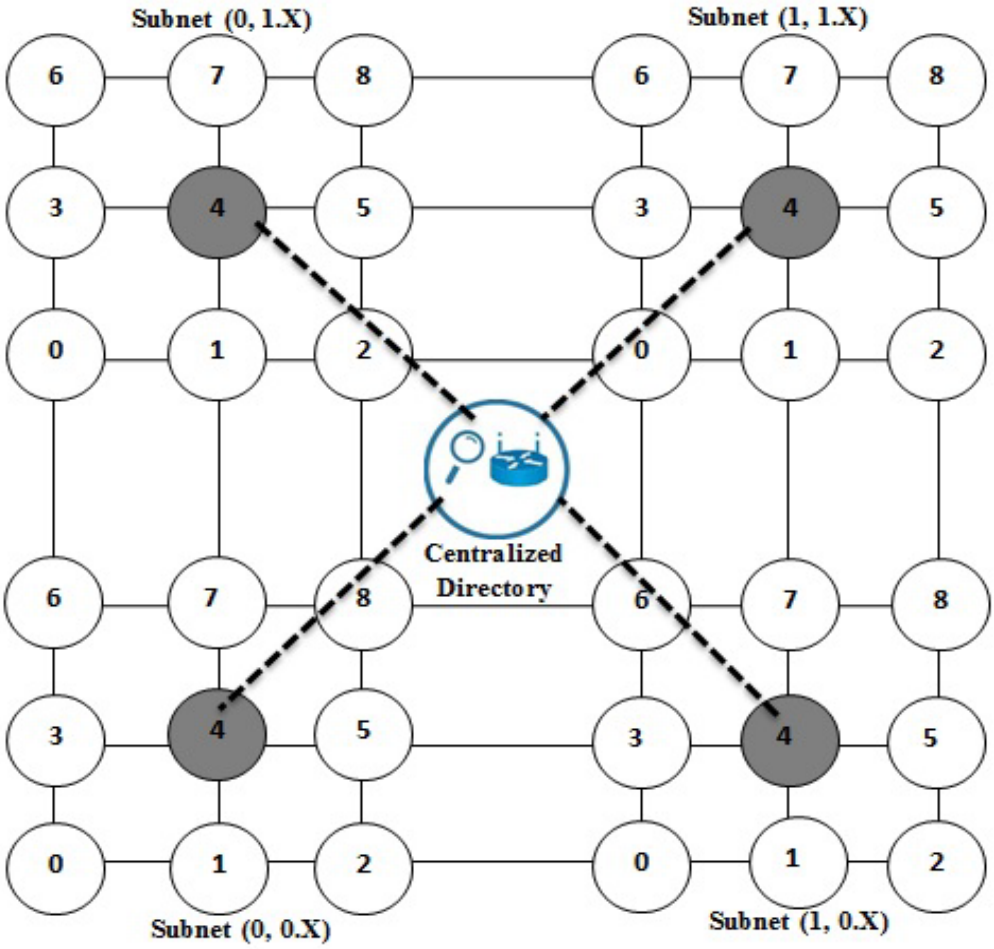


Figure 3.1: WNoC architecture with centralized directory

From the Figure 3.1, considering a 6x6 mesh topology, the cores are grouped into 3x3-core subnets, forming four quadrants. Each quadrant, that is each subnet communicates with other subnets through the centralized directory. Each dark core that is each center core supervises its own subnet and communicates with other subnets. The subnetting mechanism with the initiation of centralized directory improves the performance of the system in terms of latency, hop count, power consumption, and data synchronization.

3.3.2 Communication between Subnets with Centralized Directory

A directory is introduced in the center to hold the information of cached copies of all subnets. All the cores inside a subnet are local to the subnet and the cores outside of a subnet are remote cores for that subnet. A source core places its request for the data on the bus and if the data is not found among the caches of all the cores in the subnet, a request is sent by the subnet wireless router to the centralized directory for the requested block of data. The centralized directory has the information of all first level cache (FLC) blocks on the die. Each core may hold several memory blocks in its cache to accommodate the data fetched from the main memory.

The centralized directory updates the data information for every change in subnets and tracks the network traffic. WNoC-CD is capable of transferring packets through wired and wireless links. In WNoC-CD, processing cores are divided into various subnets which have one wireless router, responsible for providing wireless communication for the cores. The entire network is divided into subnets and each node is identified within its subnet using a local address. The address has three components (as shown in Figure 3.1 [81]) – subnet's X value, subnet's Y value, and a number for each node. Here, $X \geq 0$, $Y \geq 0$, the (X, Y) subnet specifies the subnet location in the network, and the node number identifies the processing core within the subnet. The directory is centralized and keeps track of each core. The features of addressing a specific core in a network

helps WNoC-CD provide much faster routing decisions as well as a scalable hierarchical system. In short, it's a hybrid combination of WNoC and DASH architectures. The DASH system supports directory for each cluster and the message passing technique among the clusters [58]. The minimal adaptive routing algorithm delivers shortest path and the directory maintains data sync among the cores.

However, in WNoC-CD, synchronization is complicated as it must put the requests from other cores in a queue and they are updated in a sequential order. This indicates traffic congestion and demand of bandwidth with larger capacity of directory, which makes us to think about optimizing the drawbacks of centralized directory. The distributed directories use broadcasting technique to update/sync and resolves the drawbacks of centralized directory. However, the distributed directories slightly rise power consumption for within subnet cases but reduces route time and hop counts for all cases.

3.4 Proposed Architecture 2: Introduction of Distributed Directories in WNoC Architecture with Uniform Partition of Subnets

To improvise the performance of WNoC architecture with centralized directory, distributed directories are introduced in WNoC that can manage data sync of all subnets, minimal routing path, which allows faster execution and minimal energy [82]. The proposed architecture is an improvement of the WNoC with distributed directories, wireless routers, and DASH architecture. The major goal of the proposed multicore architecture is to reduce the communication latency among the cores by decreasing the number of hops required to travel from a source core to a destination core using the distributed directories and wireless routers. The key design considerations include: communication between directories and the directory data update policy.

Unlike centralized directory, the major advantage of the distributed directories is performing the data sync by broadcasting the updates to all other directories without any waiting time.

3.4.1 Clustering Cores into Uniform Subnets with an Individual Directory

In this model, cores are divided into uniform subnets, and center core of each subnet is substituted with a directory and wireless router and is illustrated in Figure 3.2.

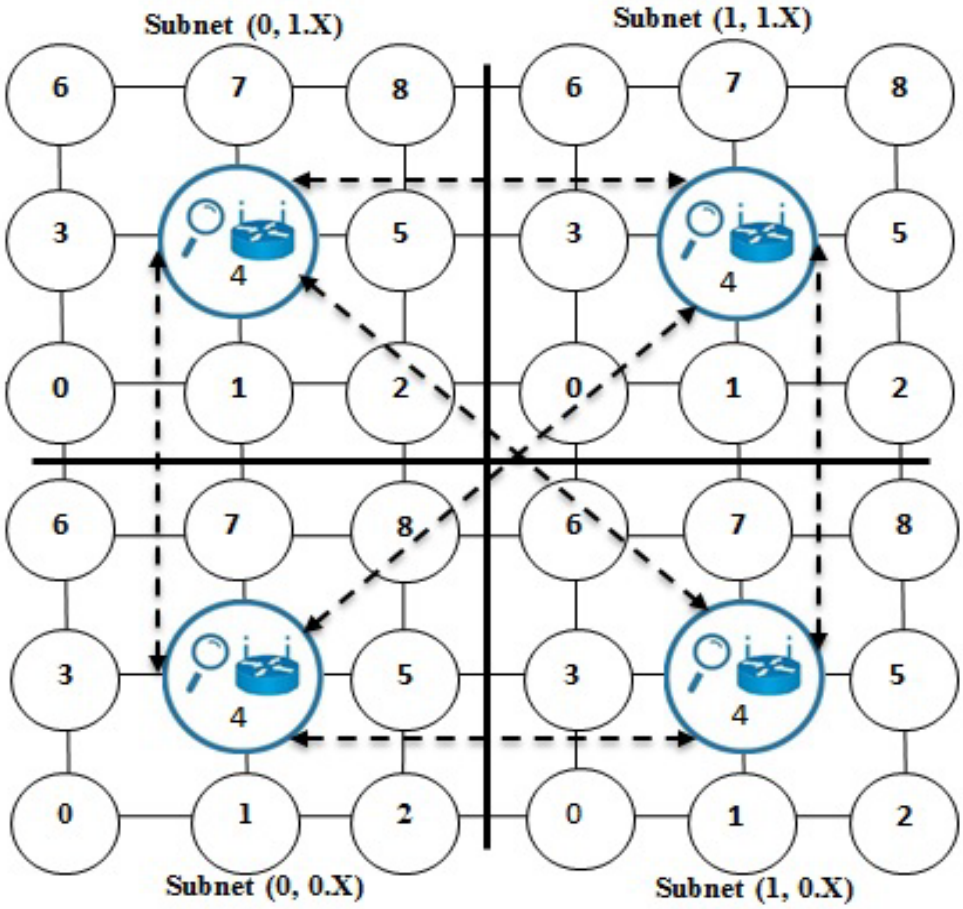


Figure3.2: WNoC architecture with distributed directories

Considering a 6x6 mesh topology, the cores are structured into 3x3-core subnets, forming four quadrants. In every subnet of Figure 3.2, Core-4 (0, 0.x) is a center core that contains a wireless router and an individual directory. The dotted line represents the wireless connections with the other subnets' center core and they are connected to one another.

3.4.2 Communication between Subnets with Distributed Directories

In this architecture, all the cores inside a subnet are local to the subnet and the cores outside of a subnet are remote cores for that subnet. The nature of communication in this design is different compared to mesh and WNoC architecture with centralized directory. Initially, a source core places its request for the data on the bus and if the data is not found among the caches of all the cores in the subnet, a request is sent by the subnet wireless router with directory to the subnet destination directory for the requested block of data. Each of the distributed directories has the information of all first level cache blocks on the die. All directories are synced to maintain data consistency. Each core may hold a few memory blocks in its cache to accommodate the data fetched from the main memory.

The directory contains the information of all other subnets that includes data sync, minimal routing path, and it is integrated with wireless router in the central core (Core-4 in Figure 3.2) of each subnet. In WNoC-DDs, all directories are identical with equal priority. The directory contains cores' subnet addresses, the status of each cached block, and the addresses of the blocks that have been cached. The directory is dynamic in nature and the total number of directory entries depends on the number of cache blocks/lines per core.

Unlike the centralized directory, the design of distributed directories reduces the pressure of accomplishing tasks on each directory. In WNoC-DDs, customized MESI protocol is used to address cache coherency. The directories with the help of customized MESI protocol can exchange data information between directories. The directories will take care of data synchronization and thus the cores are free from the role of synchronization. The directories are responsible to update their individual status and information to directories. The cores are responsible to send the information to neighbors only. Whenever the information reaches the directory, then the directory

performs the necessary operations such as sending the data to the destination core or requesting the data from any selective core. The data in and out from cores, as well as read or write data into block memory of a core is handled by the directories only.

The communication among cores inside a subnet follows mesh principle and so the delay in all three architectures is uniform and they go through the wired network. The directory is updated on every task individually. To communicate with cores in different subnets, directory and wireless routers are used. A directory is implemented in a special powerful core as illustrated in Figure 3.2 (Core-4 of each subnet). The directory quickly provides information regarding the status and address of a block cached by cores. The cores association is essential to improve communication excellence. If the destination core is physically one hop, then the cores go through wired link and finally update the directory. As a result, the communication latency among the cores is reduced significantly; this is because the source core gets the information about the destination core (i.e., requested data) quickly from the directory instead of searching other subnets.

WNoC-CD and WNoC-DDs architectures with small number of cores are not complex. With the increased number of cores, several challenges raise and the most important is the type of partition applied. Fixed subnets (i.e., uniform subnets) can't efficiently process all kinds of applications. Non-uniform subnets with distinct size of applications serve real-time workloads as they are not identical in nature. The shift in center core of subnet in non-uniform partition brings the advantage of routing with shortest hop count compared to uniform partition. To address the issues of uniform partition with large cores, non-uniform partitions are introduced. The shift of center core in non-uniform subnets reduces the multihopping drawback of uniform subnets partition.

3.5 Proposed Architecture 3: Non-Uniform Partition of Subnets in WNoC Architecture with Distributed Directories

In today's trend, multicore architecture is grabbing a full attention in commercial market as they are designed to perform better compared to traditional chip architecture. However, the bottlenecks are also accumulating and there is always a necessity to address the issues to gain more advantage on multicore or network-on-chip architectures. As the number of cores increases, they are several challenges such as, whether to assign only one task to the entire multicore CPU or assign multiple tasks to the multicore CPU. To address that, the concept of subnets that is partition of cores in WNoC has heightened. There are challenges with the fixed/uniform subnet partitioning. In general, multicore architectures are uniformly divided and each individual partition is considered as a subnet. Instead of communicating with peer cores as an entire network, the subnetwork selection allows to communicate between the subnets which will bring the latency, power consumption, and some other related parameters down when compared to non-subnet multicore architectures. Nowadays, the needs of running multiple applications are also increasing tremendously. Here comes the challenge, as the fixed subnets can only take the tasks according to the number of subnets distributed. In some cases, the number of cores required for a task or an application may be insufficient. For some applications, the cores in a subnet are more than required in a fixed subnet size, that increases the latency as well as power consumption. To address the problem of insufficient number or excess number of cores in a subnet, we examine non-uniform subnets in WNoC that should minimize latency and power consumption [83]. Non-uniform subnets have potential to bring improvement of core usage.

3.5.1 Clustering Cores into Uniform and Non-Uniform Subnets with an Individual Directory

To illustrate this approach, we consider a 64-core system with four subnets. Each subnet is segregated with 16-core and so we have four subnets. Figure 3.3 illustrates a 64-core architecture partitioned into four uniform subnets. The dark colored cores (e.g., core-9, 13, 41, and 45) are center cores with the directory and wireless router features.

Each subnet is having its own directory. Previously, results of 36-core WNoC architectures with centralized and distributed directories are discussed. In this model, centralized directory is not considered as the number of cores increased, the latency for serving a subnet request will increase and so it is evaded from the discussion.

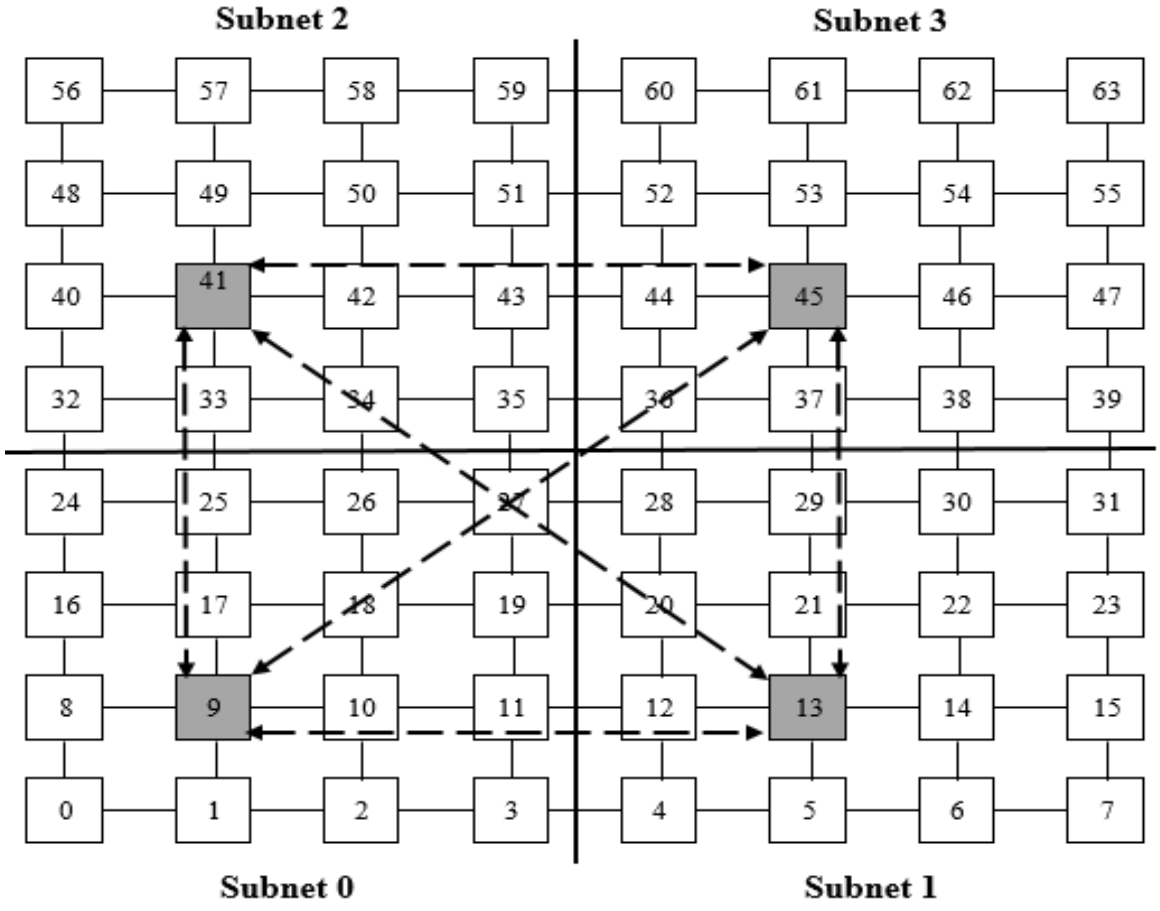


Figure 3.3: Uniform partition of subnets in 64-core architecture

- **Selection of Center Core in Even Size Subnet**

The size of a subnet is always described in row x column approach. So, m x m subnet size indicates m number of rows and m number of columns. Finding a center for even subnet size is always challenging. For example, considering Subnet 0 of Figure 3.3, the possibility of retrieving exact center core is difficult. Going closer, the opportunity for being center core is of equal priority to the cores 9, 10, 17, and 18. In this work, we are not considering any additional special cores and so we can't make the even subnet size into an odd series. Selection of any above listed cores have equal priority. There is no special reason of considering core-9 as the center core in Subnet 0. Similarly, cores 13, 41, and 45 in other subnet can be center cores. Based on specific workloads, each may offer the best performance. The dotted lines in Figure 3.3 denote wireless links between the directories with wireless routers.

- **Partitioning Cores into Non-Uniform Subnets**

Uniform subnets partition has few challenging issues and they can be addressed with an alternative way to partition subnets. Figure 3.4 illustrates the representation of non-uniform subnets of 64-core architecture and the dark colored core in each subnet is a directory with wireless router. The dotted lines in Figure 3.4 represents the wireless links between the directories. Each directory communicates with other subnet directories through wireless links.

One of the major drawbacks of uniform subnets is the latency and it depends on the type of path it follows. Every task must update the directory and so if the center directory is far from other cores, it could weak the performance of the system. Considering the drawbacks of uniform partition, subnets with assorted sizes are developed. In this way it gets benefits of assigning the subnets to different workloads. However, the non-uniform partition is fixed, and it relies on initial logical partition only.

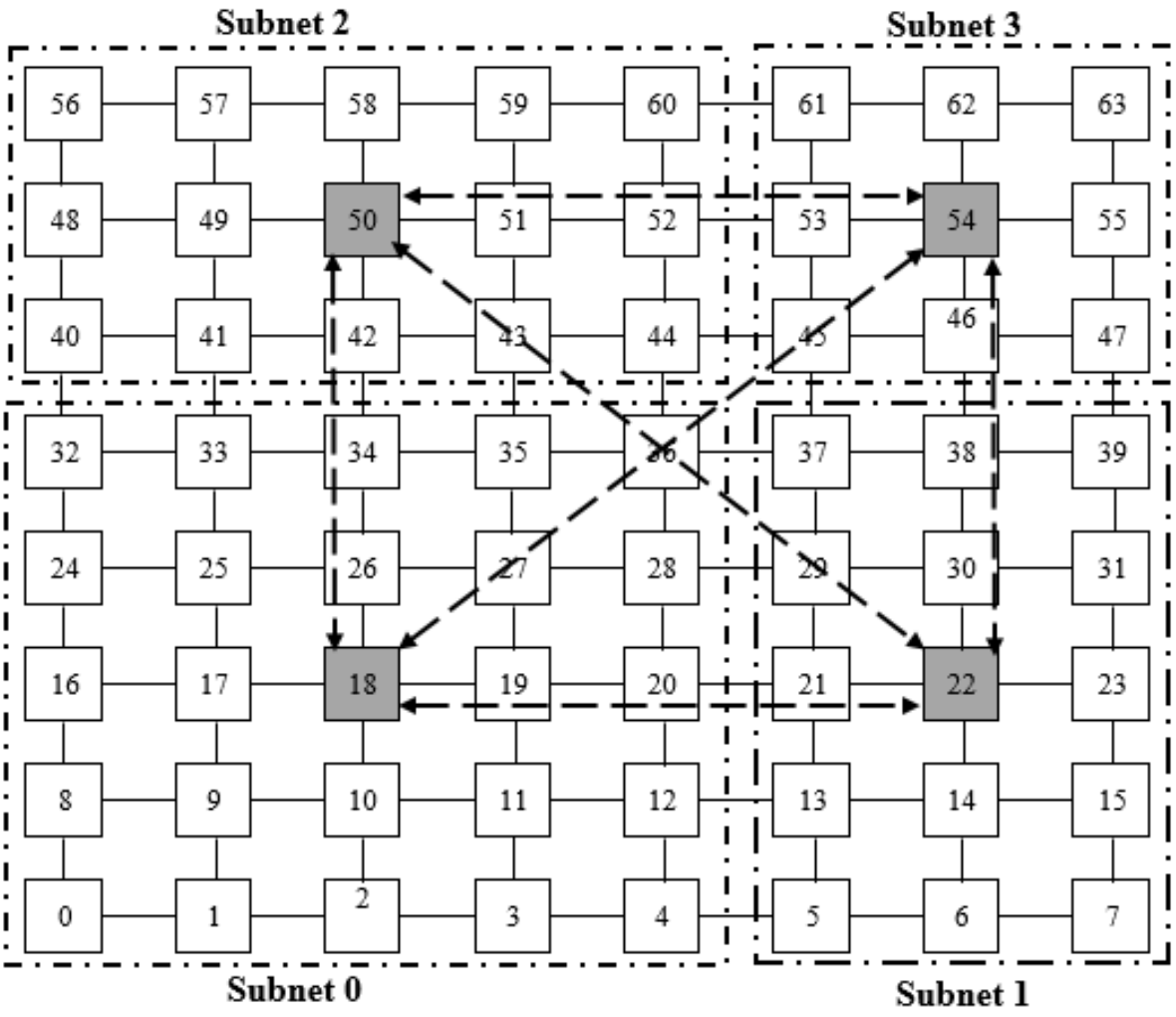


Figure 3.4: Non-uniform partition of subnets in 64-core architecture

3.5.2 Communication between Distributed Directories with Different Assignments

The communication between directories happens the same way as described for previous WNoC distributed directories architecture. However, as the size of the subnet is large, the performance improvement is not linear as several challenges rise such as wiring delays. In contrast to the previous workloads of other proposed architectures, distinctive workload with different jobs are used. Each job is sub divided into individual tasks. In this research, the jobs are given sequentially and so the waiting latency is not considered. Unlike other architectures, the path

between source to destination is not only considered between directories but the complete path of serving the request.

For every message transmission between the source and destination, unlike mesh architectures, the update of data of its subnet directory is the first or initial step. Secondly, the directories are updated through broadcasting. As there are only four directories, the interference between directories with wireless links are negotiable. Several types of jobs with individual tasks are considered where the distance between source and destination is minimum to maximum. If there is a message passing between two cores and the distance is one hop, then the destination core process the data request of source and finally the source core updates the directory. If the destination core is at a distance greater than one hop, then the source finds the route to directory of its subnet only. For in-subnet tasks, more than one hop distance, the directory after receiving the request from source, it informs the destination core to send the data directly to source core. After that the individual directory is updated and then it synchronizes other directories by broadcasting. In a similar fashion, for out-subnet tasks, directories communicate each other to find the path and status of destination core and then collecting the destination cores data through intermediate cores to source via directories. The synchronization of data through directories is faster and reduces the pressure on cores individually.

CHAPTER 4

EXPERIMENTAL DETAILS

In this chapter, we discuss experimental details to evaluate our proposed architectures. The proposed architectures are modeled using VisualSim tool. In the following sections, we discuss assumptions, tools workload used during simulation, etc.

4.1 Assumptions

The characteristics of all three architectures with wired communication are assumed as of unique behavior. Firstly, in proposed architectures 1 and 2, multicore systems with a small number of cores (36-core) are considered so that the performance of the architectures can be observed closely. Same workload is used for proposed architectures 1 and 2, to run the simulation programs of different systems. The wireless routers used in the traditional WNoC, WNoC-CD and WNoC-DDs architectures are identical. The update of directory is essential in WNoC-CD and WNoC-DDs. However, unlike WNoC-DDs, WNoC-CD need extra hops to update the directory (for some tasks) as it is centralized. For power consumption, WNoC-DDs must update through directory and thus WNoC-DDs may take more power for some tasks compared to WNoC-CD, but less power compared to mesh.

In proposed architecture 3, there are 64-core. The performance improvements with increased number of cores can be established only with the adoption of few techniques in clustering. In WNoC, subnets communicate with other subnets and cores communicate to other cores in an individual subnet to complete the data transfer requested by source core. The assumptions for the design of proposed architecture 3 is like proposed 1 and 2 architectures. The major difference is partition of subnets in non-uniform method. However, there are some additional changes in proposed architecture 3, that are used in calculating the performance

parameters such as communication latency, hop count, and power consumption. In these architectures, we calculate the complete path of requests that involves from source to destination requests and vice-versa to complete the requests of data. In the workload, source core is the one who requests the data and the destination core is the one who delivers the data to source core upon request. Mesh architecture is not considered in the non-uniform study as they are not satisfactory when compared to directory-based WNoC architectures. This is proven in the proposed 1 and proposed 2 architectures. So, the performance evaluation is only evaluated for uniform and non-uniform partition of subnets. As stated, the path considerations are different, they are explained in detail with the exploration of parameters.

- **Communication Latency**

Communication latency is a measure of time taken for transmitting a packet from source core to destination core. Communication latency depends on hop count, type of architecture and protocols used for transmission of packets. The latency is a major performance parameter, which is essential to consider in any architecture for real-time or any kind of applications [84], [85], [86]. Wormhole packet switching is considered for data delivery as it has very low transfer latency in transmitting packets. Say, a packet size of 64-bit flits is considered [87]. Where, the first flit is the header flit, which has the control information for delivering the packet to the destination address and followed by the actual payload. Intermediate nodes process just the first flit of the packet to know whether the packet is intended for itself or any other core. Only the destination core would process the whole packet. Because of that the delay caused by the intermediate nodes is less compared to the delay caused by the destination core. In an intermediate core, the delay is caused due to processing only the first flit (say, 8 Bytes). However, in a destination core, the delay is

caused due to processing the entire packet (say, 80 Bytes). Here are some of the major assumptions to calculate communication latency:

1) Delay due to an intermediate core is 4 units.
2) Delay caused due to a destination core is assumed to be 40 units.

- **Hop Count**

Hop count refers to the number of intermediate cores or routers involved for data packets transmission between source core to destination core. However, the hop count may be less or high depending upon the protocol and the type of architecture used [88], [89]. In multicore architectures with subnet mechanism, the routing methodology or algorithm differs from traditional methods and thus the number of hops differ in each architecture [90], [91]. Hop count increases if the distance between source node and destination core increases, and thus increases communication latency [92]. Here are some of the major assumptions to calculate hop count:

1) Each core is assumed to be at one hop distance from all its neighboring cores to which it is directly connected to it.
2) Wireless router cores are also assumed to be at one hop distance from its peer wireless routers.
3) Wireless router cores to the centralized directory are also assumed to be at one hop distance.
4) In WNoC-DDs, the distance between a directory to directory is also assumed as one hop.

In all three architectures, the hop count is calculated based on the number of intermediate cores, wired and wireless hops involved in successfully completing the packet transmission between source core and destination core.

- **Power Consumption**

Power consumption in all the three architectures depends on the hop count and the cores (center core needs more power) participated in the routing path for any given task. The considerations and assumptions that are made for exploring the power consumption of the three simulated architectures are listed in Table 4.1. Here are some of the major assumptions to calculate power consumption:

Table 4.1: Considerations and assumptions for power calculations

No.	Consideration	Notation	Power (Unit)
1	Power consumed by a wired link	P_{wr}	1.0
2	Power consumed by a wireless link	P_{wl}	1.1
3	Power consumed by a core with wired router	P_{cwr}	3.0
4	Core average of network- Mesh	P_{canw}	19.5
5	Average links of wired network-Mesh	P_{alwr}	5.5
6	Number of wired links	N_{wr}	(vary)
7	Number of cores wired	N_{cwr}	(vary)
8	Number of wireless links	N_{wl}	(vary)
9	Power consumed by a core with a wireless router	P_{cwl}	3.3
10	Power consumed by the wired links in a subnet on an average	P_{awrsn}	2.5
11	Power consumed between source and directory-WNoC-CD	P_{sdr}	(vary)
12	Power consumed by the directory- WNoC-CD	P_{dr}	6.0
13	Power consumed by the directory core in WNoC-CD	P_{cdr}	9.3
14	Power consumed between destination to source in WNoC-CD	P_{ds}	(vary)
15	Power consumed between source and distributed directories- WNoC-DDs	P_{sdd}	(vary)
16	Power consumed by each of the distributed directories with wireless router- WNoC-DDs	P_{ddr}	6
17	Power consumed between destination to source distributed directories- WNoC-DDs	P_{dsddr}	(vary)

- **Average of Parameters for Proposed Architectures**

The performance parameters such as communication latency, hop count, and power consumption are derived from the proposed architectures 1 and 2 by providing 25 different tasks as workload. The tasks are considered with the scenarios, that has minimum length to maximum

length between nodes. The performance of tasks can be observed individually as task wise for communication latency, hop count, and power consumption. The overall performance such as average calculation of each parameter gives precise statistics, whether to consider the new proposed architecture is beneficial compared to the other architectures. To find the decrease or improved performance of any parameter, the total column of each architecture is summed initially. The summed column of proposed architecture is subtracted from other architectures individually and finds the reduced difference.

To find the average in percentage, the ratio of reduced difference when the proposed architecture is compared with other architectures to other individual architecture summed column, and then multiplied by 100. Mathematically, it can be represented as follows:

To calculate average of parameters for n (n>1) number of tasks, when compared to mesh in %=

$$\frac{\sum_{i=1}^{n>1} Parameter\ of\ Mesh - \sum_{i=1}^{n>1} Parameter\ of\ Proposed}{\sum_{i=1}^{n>1} Parameter\ of\ Mesh} X\ 100 \quad (1)$$

To calculate average of parameters for n (n>1) number of tasks, when compared to WNoC-CD in %=

$$\frac{\sum_{i=1}^{n>1} Parameter\ of\ WNoC\ with\ CD - \sum_{i=1}^{n>1} Parameter\ of\ Proposed}{\sum_{i=1}^{n>1} Parameter\ of\ WNoC\ with\ CD}\ X\ 100 \quad (2)$$

Using Eq. (1) and Eq. (2), the average of all parameters in percentage are calculated. If we know the average, the performance of proposed can be estimated by considering the worst and best scenarios.

For proposed architecture 3, the performance of each parameter such as communication delay, hop count, and power consumption are derived for uniform and non-uniform subnets architecture with the random workloads that has 6 jobs and/or 31 individual tasks in total. The

performance of tasks can be observed individually as job basis/task basis for communication latency, hop count, and power consumption. The average computation of parameters gives overall advantage of the architectures and so it is easy to analyze the best of the architectures. The statistics of each parameter in task wise explains the best and worst condition of each architecture, which will help to make changes in design to overcome the drawbacks. To find the improved performance of any parameter, the total column of each architecture is summed initially. The summed column of proposed architecture is subtracted from other architectures individually and finds the reduced difference.

To find the average in percentage, the ratio of reduced difference when the proposed architecture is compared with other architectures to other individual architecture summed column, and then multiplied by 100. Mathematically, it can be represented as follows:

To calculate average of parameters for n (n>1) number of tasks, when compared to uniform subnets in %=

$$\frac{\sum_{i=1}^{n>1} Parameter\ of\ Uniform\ Subnets - \sum_{i=1}^{n>1} Parameter\ of\ Non-Uniform\ Subnets}{\sum_{i=1}^{n>1} Parameter\ of\ Uniform\ Subnets} X\ 100 \quad (3)$$

Using Eq. (3), the average of all parameters in percentage are calculated. If we know the average on job basis as well as task basis, the performance of proposed can be estimated and thus it is easy to identify the best and poor scenarios.

4.2 Simulation Tool

In this work, we use VisualSim Architect [93] tool to design, model, and simulate the architectures such as traditional mesh, traditional WNoC, WNoC with centralized and distributed directories, uniform and non-uniform subnets.

VisualSim Architect is a graphical modeling and simulation software for designing and validating systems that has capability of interfacing different fields such as core architectures, networking, semiconductors, real-time constraints, etc. The simulation-based results could be useful to explore different core architectures in terms of reliability, functionality, energy aware, routing protocols, and performance [94], [95]. VisualSim Architect is available with large libraries for modeling specific application components and several templates which are already modeled. The templates can be modified and extended if necessary, to explore that specific application. The engineers can model any customized design and can assemble different models using graphics editor. They can export or import data into excel, or into any graph representations. The analyzation of data could be so easy, and the statistical results will help the engineers to finalize system performance and they could check whether the demands of the system such as latency, bandwidth, power, cost, and reliability are fulfilled. Figure 4.1 illustrates the model of a subnet with a directory and wireless router. Here, the model of a single node/core with a wired router is shown in Circle A. Node '_X' Output is connected to each node and keeps track of that node assignments for all tasks. In Node '_X' Output, X represents the node number. Circle B is the model of a directory with wired and wireless router. The router is basically a switch, and the switch works according to the routing protocol. In Circle C, 'Transaction_Source' is a random workload traffic generator. The generated traffic is copied into database and so the same workload can be applied to other architectures to compare the performance improvements. The processing units help to carry out the generated traffic to the subnet and tracks the routing path of packets for every task. The routing table in Figure 4.1, has the routing information of topology that lists the routes between cores. The routing table serves as a map in delivering the packets to the destination cores.

The destination node may be directly connected to source node or it might be routed via other nodes.

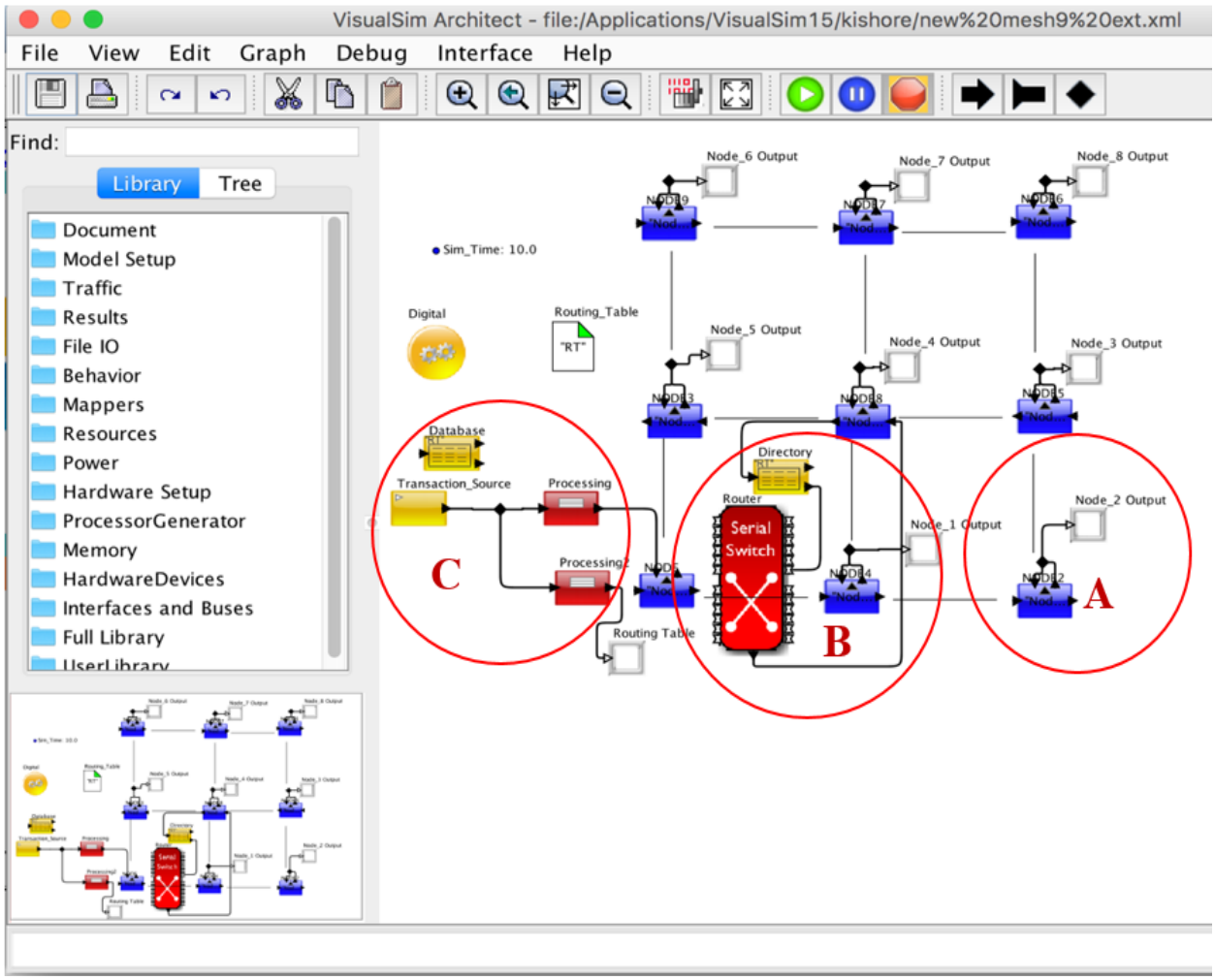


Figure 4.1: Model of the subnet with a directory and wireless router

VisualSim Architect is advantageous because of its architecture modeling capabilities as they are useful in prior to the physical implementation of system design or algorithms. This modeling helps in time saving, useful to finalize whether the system is practically implementable, pros and cons of designs, trade-offs between different architectures or technologies, validation of modeled designs and many other features can be derived. In short, VisualSim Architect allows us to formalize the system specifications by providing organized and quantitative solutions.

4.3 Workload

Workload is used in the experiments to run the simulation programs and calculate latency, hop count, and power consumption. In this work, workload for proposed architectures 1 and 2 is the same and so the performance improvements can be observed clearly. This method allows us to closely observe the performance improvement in each task. In this way, the advancements in architecture can be analyzed and it can convey the followed procedure is acceptable or not acceptable. In proposed architecture 3 with 64-core, a complex workload is considered to evaluate the performance comparison through application basis with distinct subtasks load.

- **Workload for Proposed Architectures 1 and 2**

In this work, we consider 25 different tasks as illustrated in Table 4.2 for the proposed architectures 1 and 2. The tasks 1 to 20 represent the requests from one subnet to other and the cases 21 to 25 represents the requests from one node to another node within the same subnet. The workload to represent the communications between source nodes and destination nodes are generated using VisualSim Architect tool.

Table 4.2: Source and destination cores for different communication tasks

Different Scenarios	Source Core (S)	Destination Core (D)	Subnet Location
Task 1	Core - > (0, 0.0)	Core - > (1, 1.8)	Out-Subnet
Task 2	Core - > (0, 0.4)	Core - > (1, 1.4)	Out-Subnet
Task 3	Core - > (0, 0.7)	Core - > (1, 0.1)	Out-Subnet
Task 4	Core - > (0, 0.3)	Core - > (0, 1.5)	Out-Subnet
Task 5	Core - > (1, 0.5)	Core - > (0, 1.2)	Out-Subnet
Task 6	Core - > (1, 0.7)	Core - > (0, 1.5)	Out-Subnet
Task 7	Core - > (0, 1.0)	Core - > (1, 0.0)	Out-Subnet
Task 8	Core - > (0, 0.8)	Core - > (1, 1.6)	Out-Subnet
Task 9	Core - > (0, 0.7)	Core - > (0, 1.1)	Out-Subnet
Task 10	Core - > (1, 1.5)	Core - > (1, 0.2)	Out-Subnet
Task 11	Core - > (0, 1.3)	Core - > (0, 0.1)	Out-Subnet
Task 12	Core - > (0, 1.4)	Core - > (1, 0.6)	Out-Subnet
Task 13	Core - > (1, 0.1)	Core - > (1, 1.1)	Out-Subnet

Table 4.2 (continued)

Different Scenarios	Source Core (S)	Destination Core (D)	Subnet Location
Task 14	Core - > (1, 1.2)	Core - > (0, 0.8)	Out-Subnet
Task 15	Core - > (1, 0.6)	Core - > (0, 1.2)	Out-Subnet
Task 16	Core - > (1, 0.4)	Core - > (0, 1.7)	Out-Subnet
Task 17	Core - > (1, 1.3)	Core - > (0, 1.3)	Out-Subnet
Task 18	Core - > (0, 1.2)	Core - > (1, 1.0)	Out-Subnet
Task 19	Core - > (0, 0.1)	Core - > (1, 0.7)	Out-Subnet
Task 20	Core - > (1, 0.2)	Core - > (0, 1.6)	Out-Subnet
Task 21	Core - > (0, 0.6)	Core - > (0, 0.5)	In-Subnet
Task 22	Core - > (1, 0.7)	Core - > (1, 0.8)	In-Subnet
Task 23	Core - > (0, 1.4)	Core - > (0, 1.2)	In-Subnet
Task 24	Core - > (1, 1.6)	Core - > (1, 1.2)	In-Subnet
Task 25	Core - > (0, 1.7)	Core - > (0, 1.1)	In-Subnet

- **Workload for Proposed Architecture 3**

Unlike 36-core architecture, a different workload is considered for the 64-core architecture. This workload allows us to compute the performance in different parameters according to job and individual task basis. The details of the workload are listed in Table 4.3.

In this workload, tasks are included with in-subnet scenarios and out-subnet scenarios. The performance evaluation of the architecture is derived on job basis as well as individual task basis. The jobs are given sequentially and are serviced according to the request order. Here the jobs are not identical, where they differ in number of tasks and location of subnets that is in or out.

To evaluate the best of the architectures, random tasks are generated where few tasks may give advantage to uniform subnets and some other to non-uniform subnet partition. Non-uniform subnet is a trade-off approach for large core architectures like more than 64-core. The method of proposed architecture 3 can be extended to any large number of cores. The random scenarios are generated using VisualSim tool for jobs, with an instruction to consider in or out-subnets.

Table 4.3: Workload for uniform and non-uniform subnets in 64-core architecture

Different Scenarios	Subtasks between Cores	Uniform Partition		Non-Uniform Partition	
		Subnets Involved	Subnet Location	Subnets Involved	Subnet Location
Job 1	18-54	S0, S3	Out	S0, S3	Out
	59-19	S3, S0	Out	S2, S0	Out
	19-51	S0, S3	Out	S0, S2	Out
	18-50	S0, S2	Out	S0, S2	Out
	58-26	S2, S0	Out	S2, S0	Out
Job 2	19-20	S0,S1	Out	S0	In
	60-51	S3,S2	Out	S2	In
	52-50	S3,S2	Out	S2	In
	24-20	S0,S1	Out	S0	In
Job 3	6-28	S1	In	S1, S0	Out
	31-20	S1	In	S1, S0	Out
	63-39	S3	In	S3, S1	Out
	59-35	S2	In	S2, S0	Out
Job 4	19-35	S0, S2	Out	S0	In
	17-34	S0, S2	Out	S0	In
	38-14	S3, S1	Out	S1	In
	49-52	S2, S3	Out	S2	In
	23-20	S1	In	S1	Out
	4-31	S1	In	S1	Out
	60-39	S3	In	S2, S1	Out
Job 5	54-63	S3	In	S3	In
	53-55	S3	In	S3	In
	47-61	S3	In	S3	In
	63-62	S3	In	S3	In
	53-45	S3	In	S3	In
	46-62	S3	In	S3	In
	55-47	S3	In	S3	In
Job 6	9-45	S0, S3	Directory	S0, S3	Out
	54-50	S3, S2	Out	S3, S2	Directory
	18-22	S1	Out	S0, S1	Directory
	13-41	S1,S3	Directory	S1, S2	Out

4.4 Simulation of Proposed Architecture 1

In this architecture, the performance is compared by assigning the unique workload for the architectures like mesh, traditional WNoC, and proposed WNoC with centralized directory (WNoC-CD). In proposed architecture 1, with the introduction of directory, traditional method of

multiple routing in mesh topology, and traffic as well as broadcasting issues of traditional WNoC can be avoided. Data synchronization is easy with the directories as they are having proven history to address cache coherence and scaling issues when compared to snoopy protocols. The performance is observed through randomly assigned workloads and the parameters considered are communication latency, hop count, and power consumption.

4.4.1 Communication Latency

The communication latency of an architecture depends on their routing methodology. The information from source to destination flows through intermediate nodes. In mesh multicasting, XY routing algorithm is followed which is an orthodox strategy and that can eventually lead to a longer delay, especially for the end-to-end communications. The information is generally transmitted in packets that have header, payload and trailer. Tasks 1 through 20 can be better executed by WNoC architecture. In traditional WNoC, the routing to destination is primarily checked within subnet and if the address is not in the subnet, then it broadcasts the same information to all other subnets. While broadcasting, the communication is through wireless routers, so it has the possibility of skipping the unnecessary intermediate nodes and thus reduces the latency. Even though the destination is just one hop away from its subnet, WNoC will follow the broadcasting methodology and it may take longer path compared to the mesh multicasting in few tasks.

As illustrated in Table 4.4, for some tasks (such as Tasks 8 and 9) traditional WNoC takes more time than mesh, for some tasks (such as Task 10) traditional WNoC and mesh take same amount of time, and for some tasks (such as Tasks 1, 2, and 20) traditional WNoC takes less time than mesh. However, for all the tasks WNoC-CD takes less or equal time compared to traditional mesh and traditional WNoC architectures.

If the destination is only one hop distance (Tasks such as 9, 18 and 22), then all the networks behave as mesh and the communication latency is identical in mesh and WNoC-CD, but traditional WNoC takes additional latency to update the subnets.

Table 4.4: Communication latency compared to WNoC-CD architecture

Different Scenarios	**Traditional Mesh (ms)**	**Traditional WNoC (ms)**	**Proposed WNoC-CD (ms)**
Task 1: (0,0.0)-(1,1.8)	4x9+40=76	4x4+40=56	4x2+40=48
Task 2: (0,0.4)-(1,1.4)	4x5+40=60	4x0+40=40	4x0+40=40
Task 3: (0,0.7)-(1,0.1)	4x4+40=56	4x2+40=48	4x1+40=44
Task 4: (0,0.3)-(0,1.5)	4x4+40=56	4x2+40=48	4x1+40=44
Task 5: (1,0.5)-(0,1.2)	4x4+40=56	4x3+40=52	4x1+40=44
Task 6: (1,0.7)-(0,1.5)	4x3+40=52	4x2+40=48	4x1+40=44
Task 7: (0,1.0)-(1,0.0)	4x5+40=60	4x4+40=56	4x2+40=48
Task 8: (0,0.8)-(1,1.6)	4x3+40=52	4x4+40=56	4x2+40=48
Task 9: (0,0.7)-(0,1.1)	4x0+40=40	4x2+40=48	4x0+40=40
Task 10: (1,1.5)-(1,0.2)	4x3+40=52	4x3+40=52	4x1+40=44
Task 11: (0,1.3)-(0,0.1)	4x4+40=56	4x2+40=48	4x1+40=44
Task 12: (0,1.4)-(1,0.6)	4x3+40=52	4x2+40=48	4x0+40=40
Task 13: (1,0.1)-(1,1.1)	4x2+40=48	4x2+40=48	4x1+40=44
Task 14: (1,1.2)-(0,0.8)	4x3+40=52	4x4+40=56	4x2+40=48
Task 15: (1,0.6)-(0,1.2)	4x1+40=44	4x4+40=56	4x2+40=48
Task 16: (1,0.4)-(0,1.7)	4x6+40=64	4x1+40=44	4x0+40=40
Task 17: (1,1.3)-(0,1.3)	4x2+40=48	4x2+40=48	4x1+40=44
Task 18: (0,1.2)-(1,1.0)	4x0+40=40	4x4+40=56	4x0+40=40
Task 19: (0,0.1)-(1,0.7)	4x4+40=56	4x2+40=48	4x1+40=44
Task 20: (1,0.2)-(0,1.6)	4x9+40=76	4x4+40=56	4x2+40=48
Task 21: (0,0.6)-(0,0.5)	4x2+40=48	4x2+40=48	4x2+40=48
Task 22: (1,0.7)-(1,0.8)	4x0+40=40	4x0+40=40	4x0+40=40
Task 23: (0,1.4)-(0,1.2)	4x1+40=44	4x1+40=44	4x1+40=44
Task 24: (1,1.6)-(1,1.2)	4x3+40=52	4x3+40=52	4x3+40=52
Task 25: (0,1.7)-(0,1.1)	4x1+40=44	4x1+40=44	4x1+40=44

The detailed explanation of Table 4.4 for each architecture can be better known by discussing with any task. Let's consider the Task 1, which is the maximum distance between source and destination cores. The information is generally transmitted in packets that have header, payload and trailer. Here the header size, say 8 bytes and the whole packet is 80 bytes. Therefore,

if the delay due to an intermediate core is four units, the delay caused due to a destination core is assumed to be 40 units. The intermediate cores check only the header flit and so each intermediate core causes four units of delay. In mesh, for Task 1, they are nine intermediate cores and one destination core excluding source core. So, delay due to nine intermediate cores will be 36 (4*9=36) units and the destination core takes 40 units, which will make the total as 76 units. In WNoC-CD, the centralized directory is considered as destination core. So, in Task 1, it has two intermediate cores and one destination core (centralized directory) involved. In detail, delay due to intermediate cores is 8 (4*2=8) units and the destination core takes 40 units, which will make the total as 48 units.

4.4.2 Hop Count

Hop count is another significant performance characteristic to ensure the architecture could be faster with reliable communication. To determine hop count, the number of hops involved in data transmission are counted and they differ based on the architecture. The calculation of hop count for each task is illustrated in Table 4.5.

In some scenarios, even though the hop count is less, it may not ensure faster communication or data transmission. This is because the communication latency also varies based on selected path, such as single path with higher number of hops or multiple paths with low hop count. Generally, multiple paths may cause more delay as intermediate routers or devices may take long time for processing the data transmission. In most cases, if the packet exceeds the large hop count for a network, then that packet is discarded and there should be a retransmission of packet to accomplish successful task completion. To ensure that the communication is successful in mesh architecture, return path or acknowledgement is essential. So, the total number of hops (H_T) is 2x the number of hops between the source and destination.

Table 4.5: Hop count compared to WNoC-CD architecture

Different Scenarios	**Traditional Mesh**	**Traditional WNoC**	**Proposed WNoC-CD**
Task 1: (0,0.0)-(1,1.8)	HC= H_T *2 (S to D)+ H_T *2 (D to S)=20+20=40	HC= H_T *2 (S to D)+ H_T *2 (D to S)=10+10=20	HC= H_T (S to Directory)+ H_T (D to S) =3+6=9
Task 2: (0,0.4)-(1,1.4)	HC=12+12=24	HC=2+2=4	HC=1+2=3
Task 3: (0,0.7)-(1,0.1)	HC=10+10=20	HC=6+6=12	HC=2+4=6
Task 4: (0,0.3)-(0,1.5)	HC=10+10=20	HC=6+6=12	HC=2+4=6
Task 5: (1,0.5)-(0,1.2)	HC=10+10=20	HC=8+8=16	HC=2+5=7
Task 6: (1,0.7)-(0,1.5)	HC=8+8=16	HC=6+6=12	HC=2+4=6
Task 7: (0,1.0)-(1,0.0)	HC=12+12=24	HC=10+10=20	HC=3+6=9
Task 8: (0,0.8)-(1,1.6)	HC=8+8=16	HC=10+10=20	HC=3+6=9
Task 9: (0,0.7)-(0,1.1)	HC=2+2=4	HC=6+6=12	HC=1+1+2=4
Task 10: (1,1.5)-(1,0.2)	HC=8+8=16	HC=8+8=16	HC=2+5=7
Task 11: (0,1.3)-(0,0.1)	HC=10+10=20	HC=6+6=12	HC=2+4=6
Task 12: (0,1.4)-(1,0.6)	HC=8+8=16	HC=6+6=12	HC=1+4=5
Task 13: (1,0.1)-(1,1.1)	HC=6+6=12	HC=6+6=12	HC=2+4=6
Task 14: (1,1.2)-(0,0.8)	HC=8+8=16	HC=10+10=20	HC=3+6=9
Task 15: (1,0.6)-(0,1.2)	HC=4+4=8	HC=10+10=20	HC=3+6=9
Task 16: (1,0.4)-(0,1.7)	HC=14+14=28	HC=4+4=8	HC=1+3=4
Task 17: (1,1.3)-(0,1.3)	HC=6+6=12	HC=6+6=12	HC=2+4=6
Task 18: (0,1.2)-(1,1.0)	HC=2+2=4	HC=10+10=20	HC=1+1+3=5
Task 19: (0,0.1)-(1,0.7)	HC=10+10=20	HC=6+6=12	HC=2+4=6
Task 20: (1,0.2)-(0,1.6)	HC=20+20=40	HC=10+10=20	HC=3+6=9
Task 21: (0,0.6)-(0,0.5)	HC=6+6=12	HC=6+6=12	HC=3+3=1=7
Task 22: (1,0.7)-(1,0.8)	HC=2+2=4	HC=2+2=4	HC=1+1+2=4
Task 23: (0,1.4)-(0,1.2)	HC=4+4=8	HC=4+4=8	HC=2+2+1=5
Task 24: (1,1.6)-(1,1.2)	HC=8+8=16	HC=8+8=16	HC=4+4+1=9
Task 25: (0,1.7)-(0,1.1)	HC=4+4=8	HC=4+4=8	HC=2+2+1=5

However, the proposed architecture does not require any acknowledgement path for identifying the status as well as fetching the data. The routing path has become straightforward and less due to the introduction of directory in a multicore architecture. The directory works as a commander and supervises the purpose without any acknowledgement.

The detailed explanation of any task for each architecture can be simplified by considering a task. Let's consider Task 1, which has maximum end-to-end communication. In mesh, to

communicate between source and destination core it has 10 intermediate hops. Usually in mesh, it should get an acknowledgement to send any information. So, it has double path for source and destination which makes 20 hop counts. Similarly, to acknowledge the information is completely received from destination to source is also double which makes 20 hop count and so in total it has 40 hop counts. In WNoC-CD, the request to fetch data is up to centralized directory that is three hops and then the return path is from destination to source core that is six hops, which makes the total as nine hops. In WNoC-DDs, the request to fetch is to its individual directory only as the directories are synced that takes two hops, and then the return path is five hops which makes the total as seven hops.

4.4.3 Power Consumption

To calculate the power (assumptions in Table 4.1) consumed for a task, there are several considerations such as cores, routers, and directories involved in reaching destination core from source core. It is assumed that each wired link consumes one unit of power (P_{wr}). Studies indicate a wired network connection would take less power than a wireless network [96], [97], [98], [99]. Therefore, a wireless link is assumed to consume 1.1 unit of power (P_{wl}). To be in the conservative side, we assume that a core with wired router consumes three units of power (P_{cwr}). The XY routing algorithm [100], [101], [102] does not have a unique pattern path towards destination. So, the average power consumed by a core in a 6x6 mesh (P_{canw}) is average number of cores travelled multiplied by power needed for each wired core (19.5 units). Similarly, the average power consumed by a wired link in a 6x6 mesh (P_{alwr}) is 5.5 power units. The power consumption of each individual task can be observed in Table 4.6.

Table 4.6: Power consumption compared to WNoC-CD architecture

Different Scenarios	**Traditional Mesh (mW)**	**Traditional WNoC (mW)**	**Proposed WNoC-CD (mW)**
Task 1: (0,0.0)-(1,1.8)	P_1=24, P_2=24, P_3=25, P_{tot}=73	P_{sd}=37.6, P_{ds}=24.7 P_{tot}=62.3	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2
Task 2: (0,0.4)-(1,1.4)	P_1=24, P_2=24, P_3=25, P_{tot}=73	P_{sd}=37.6, P_{ds}=24.7 P_{tot} =62.3	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2
Task 3: (0,0.7)-(1,0.1)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sd}=31.6, P_{ds}=12.7 P_{tot} =44.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 4: (0,0.3)-(0,1.5)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 5: (1,0.5)-(0,1.2)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sd}=34.6, P_{ds}=21.7 P_{tot} =56.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 6: (1,0.7)-(0,1.5)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 7: (0,1.0)-(1,0.0)	P_1=27, P_2=27, P_3=25 P_{tot}=79	P_{sd}=37.6, P_{ds}=24.7 P_{tot} =62.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2
Task 8: (0,0.8)-(1,1.6)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd}=37.6, P_{ds}=24.7 P_{tot} =62.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2
Task 9: (0,0.7)-(0,1.1)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 10: (1,1.5)-(1,0.2)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd}=34.6, P_{ds}=21.7 P_{tot} =56.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 11: (0,1.3)-(0,0.1)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 12: (0,1.4)-(1,0.6)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd}=31.6, P_{ds}=18.7 P_{tot}=50.3	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2
Task 13: (1,0.1)-(1,1.1)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 14: (1,1.2)-(0,0.8)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd}=37.6, P_{ds}=24.7 P_{tot}=62.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2
Task 15: (1,0.6)-(0,1.2)	P_1=11, P_2=11, P_3=25 P_{tot}=47	P_{sd}=37.6, P_{ds}=24.7 P_{tot}=62.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2
Task 16: (1,0.4)-(0,1.7)	P_1=31, P_2=31, P_3=25 P_{tot}=87	P_{sd}=37.6, P_{ds}=24.7 P_{tot}=62.3	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2
Task 17: (1,1.3)-(0,1.3)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sd}=37.6, P_{ds}=24.7 P_{tot} =62.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 18: (0,1.2)-(1,1.0)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sd}=31.6, P_{ds}=12.7 P_{tot} =44.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2
Task 19: (0,0.1)-(1,0.7)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sd}=34.6, P_{ds}=18.7 P_{tot} =53.3	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2
Task 20: (1,0.2)-(0,1.6)	P_1=43, P_2=43, P_3=25 P_{tot}=111	P_{sd}=34.6, P_{ds}=21.7 P_{tot} =56.3	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2

Table 4.6 (continued)

Different Scenarios	Traditional Mesh (mW)	Traditional WNoC (mW)	Proposed WNoC-CD (mW)
Task 21: (0,0.6)-(0,0.5)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sd}=14.5, P_{ds}=14.5 P_{tot}=29	P_{sdr}=15.9, P_{ds}=14.8 P_{tot}=30.7
Task 22: (1,0.7)-(1,0.8)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sd} =8.5, P_{ds}=8.5 P_{tot}=17	P_{sdr}=12.9, P_{ds}=8.5 P_{tot}=21.4
Task 23: (0,1.4)-(0,1.2)	P_1=11, P_2=11, P_3=25 Ptot=47	P_{sd} =11.8, P_{ds}=11.8 P_{tot}=23.6	P_{sdr}=12.9, P_{ds}=11.8 P_{tot}=24.7
Task 24: (1,1.6)-(1,1.2)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sd} =17.5, P_{ds}=17.5 P_{tot}=35	P_{sdr}=18.9, P_{ds}=17.8 P_{tot}=36.7
Task 25: (0,1.7)-(0,1.1)	P_1=11, P_2=11, P_3=25 P_{tot}=47	P_{sd} =11.8, P_{ds}=11.8 P_{tot}=23.6	P_{sdr}=12.9, P_{ds}=11.8 P_{tot}=24.7

For the WNoC-CD architecture, each wired core consumes three units of power (like a core in mesh) and the special core with wireless router consumes 3.3 units of power (P_{cwl}). For a 3x3 subnet, the minimum number of links is one and the maximum number of links is four. So, the average power consumed by a wired link in a subnet (P_{awrsn}) is 2.5 power units.

The power consumed by the directory (P_{dr} = 6 units) is assumed to be twice the power consumed by the core, since the entire directory must be scanned in a worst scenario. Similarly, the power consumed by the central directory-core with wireless router ($P_{cdr} = P_{dr}$ + power needed by a wireless core = 6 + 3.3) is 9.3. The following is an example for calculating power consumption by considering Task 1 (0,0.0)-(1,1.8):

<u>In Mesh multicasting:</u>

$P_1 = (P_{wr}*N_{wr})+(P_{cwr}*N_{cwr}) = 43$
$P_2 = (P_{wr}*N_{wr})+(P_{cwr}*N_{cwr}) = 43$
$P_3 = P_{alwr}+P_{canw} = 5.5 + 19.5 = 25$
$P_{tot} = P_1 + P_2 + P_3 = 43 + 43 + 25 = 111$

<u>In WNoC:</u>

$P_{sd}=P_{awrsn}+(P_{cwr}*N_{cwr})+P_{cwl}+3(P_{wl}+P_{casn}) = 2.5+(3*2)+3.3+25.8 = 37.6$
$P_{ds} = P_{dsn}+P_{wl}+P_{ssn} = 11.8+1.1+11.8 = 24.7$
$P_{tot} = P_{sd} + P_{ds} = 37.6+24.7=62.3$

In WNoC-CD:

$P_{tot} = P_{sdr} + P_{cdr}$ (For Out-Subnet)
$P_{sdr}=P_{awrsn}+(P_{cwr}*N_{cwr})+P_{cwl}+P_{wl} = 2.5+(3*2)+3.3+1.1=12.9$
$P_{cdr}= P_{dr}+P_{cwl}= 6 + 3.3 = 9.3$
$P_{tot}=12.9+9.3=22.2$
$P_{ds}=P_{awrsn}+(P_{cwr}*N_{cwr})+P_{cwl}$ (For In-Subnet)

4.5 Simulation of Proposed Architecture 2

In this architecture, the performance is compared by assigning the unique workload for the architectures like traditional mesh, WNoC-CD, and proposed WNoC with distributed directories (WNoC-DDs). In proposed architecture 2, with the introduction of the directory in each subnet may improve the performance as a significant factor. In each subnet, directory is added to the center core that has wireless router. As the directory is synced at every instant of tasks to the cores, the data transfer is easy and faster. The directories transfer the data from the destination core to the requested source core. Source cores in proposed WNoC-DDs architecture from any subnet, need not to wait for other subnet cores as in WNoC-CD. In WNoC-CD, the requests from the subnets can be executed sequentially as the directory is only one and it is the only medium of communication. With the introduction of directory in each subnet, waiting time for directory is terminated and so the tasks can be executed in parallel without the intervention of other subnets. However, the directory synchronization may take additional time for an update from a subnet, but it is very less when compared to waiting time in WNoC-CD architecture.

4.5.1 Communication Latency

In proposed architecture 2, the calculation of communication latency is like the proposed architecture 1, which depends on their routing methodology. However, the major difference is the way of data aggregating to the requested cores. In all these architectures, source core is the one who requests data and the destination core is the one who sends the data. In proposed WNoC-DDs,

the communication is also subnet to subnet like proposed architecture 1 (WNoC-CD) but the difference is the subnet is supervised with an individual directory which is integrated with wireless router. The directory adds more intelligence in data transfer compared to WNoC-CD. Each directory is holding the data of other subnets and so it reduces the latency at the cost of broadcasting to all directories for every task execution. Table 4.7 illustrates the communication latency of all the architectures and the performance can be observed for each task.

Table 4.7: Communication latency compared to WNoC-DDs architecture

Different Scenarios	**Traditional Mesh (ms)**	**WNoC-CD (ms)**	**Proposed WNoC-DDs (ms)**
Task 1: (0,0.0)-(1,1.8)	4x9+40=76	4x2+40=48	4x1+40=44
Task 2: (0,0.4)-(1,1.4)	4x5+40=60	4x0+40=40	4x0+40=40
Task 3: (0,0.7)-(1,0.1)	4x4+40=56	4x1+40=44	4x0+40=40
Task 4: (0,0.3)-(0,1.5)	4x4+40=56	4x1+40=44	4x0+40=40
Task 5: (1,0.5)-(0,1.2)	4x4+40=56	4x1+40=44	4x0+40=40
Task 6: (1,0.7)-(0,1.5)	4x3+40=52	4x1+40=44	4x0+40=40
Task 7: (0,1.0)-(1,0.0)	4x5+40=60	4x2+40=48	4x1+40=44
Task 8: (0,0.8)-(1,1.6)	4x3+40=52	4x2+40=48	4x1+40=44
Task 9: (0,0.7)-(0,1.1)	4x0+40=40	4x0+40=40	4x0+40=40
Task 10: (1,1.5)-(1,0.2)	4x3+40=52	4x1+40=44	4x0+40=40
Task 11: (0,1.3)-(0,0.1)	4x4+40=56	4x1+40=44	4x0+40=40
Task 12: (0,1.4)-(1,0.6)	4x3+40=52	4x0+40=40	4x0+40=40
Task 13: (1,0.1)-(1,1.1)	4x2+40=48	4x1+40=44	4x0+40=40
Task 14: (1,1.2)-(0,0.8)	4x3+40=52	4x2+40=48	4x1+40=44
Task 15: (1,0.6)-(0,1.2)	4x1+40=44	4x2+40=48	4x1+40=44
Task 16: (1,0.4)-(0,1.7)	4x6+40=64	4x0+40=40	4x0+40=40
Task 17: (1,1.3)-(0,1.3)	4x2+40=48	4x1+40=44	4x0+40=40
Task 18: (0,1.2)-(1,1.0)	4x0+40=40	4x0+40=40	4x0+40=40
Task 19: (0,0.1)-(1,0.7)	4x4+40=56	4x1+40=44	4x0+40=40
Task 20: (1,0.2)-(0,1.6)	4x9+40=76	4x2+40=48	4x1+40=44
Task 21: (0,0.6)-(0,0.5)	4x2+40=48	4x2+40=48	4x2+40=48
Task 22: (1,0.7)-(1,0.8)	4x0+40=40	4x0+40=40	4x0+40=40
Task 23: (0,1.4)-(0,1.2)	4x1+40=44	4x1+40=44	4x1+40=44
Task 24: (1,1.6)-(1,1.2)	4x3+40=52	4x3+40=52	4x3+40=52
Task 25: (0,1.7)-(0,1.1)	4x1+40=44	4x1+40=44	4x1+40=44

WNoC-DDs perform better as they reduce the intermediate cores in performing data transfer between source to destination. WNoC-DDs follow the adaptive XY routing algorithm to transfer data between cores and it is advantageous as it searches for alternative paths if the traffic is high at the intermediate cores. The worst scenarios of mesh multicasting (such as end-to-end communication) and one hop away between two subnets scenarios of WNoC can be avoided in WNoC-DDs architecture. WNoC-DDs should take less time in all those scenarios. The detailed statistics of the use of the subnets is maintained and monitored by the directory. The destination cores are considered based on the activities of the subnets. Thus, the directory should help balance load by selecting the destination cores from different subnets (if possible). However, the routing path to communicate within the subnet (Tasks 21 to 25) is the same and so the delay is unique for all the three architectures namely traditional mesh, WNoC-CD, and proposed WNoC-DDs and is illustrated in Table 4.7.

WNoC-DDs takes less time due to the introduction of distributed directories. The directories sync the data of their own subnet as well as other subnets through neighbor directories by using customized MESI protocol. As the directories are synced, they avoid broadcasting issues as well as bandwidth issues. So, when the source reaches its own subnet directory then it could be considered as it reached the destination. In WNoC-DDs, the individual directory is considered as destination. So, in Task 1 (0,0.0)-(1,1.8), it has only one intermediate core that takes four units and one destination (directory) core that takes 40 units, which will make the total as 44 units whereas the traditional mesh takes 76 units and WNoC-CD takes 48 units.

4.5.2 Hop Count

Hop count determines the number of hops involved in transferring data between source and destination. The performance can be higher if the number of hops reduced. In WNoC-DDs, the

hops are reduced as they skip the intermediate cores and mostly receives the data through its own subnet directory with minimal latency compared to WNoC-CD and other architectures. The number of hops involved for data transmission in each task is considered as hop count. The calculation of hop count for each task is illustrated in Table 4.8.

Table 4.8: Hop count compared to WNoC-DDs architecture

Different Scenarios	**Traditional Mesh**	**WNoC-CD**	**Proposed WNoC-DDs**
Task 1: (0,0.0)-(1,1.8)	HC= H_T *2 (S to D)+ H_T *2 (D to S)=20+20=40	HC= H_T (S to Directory)+ H_T (D to S) =3+6=9	HC= H_T (S to Directory)+ H_T (D to S) =2+5=7
Task 2: (0,0.4)-(1,1.4)	HC=12+12=24	HC=1+2=3	HC=0+1=1
Task 3: (0,0.7)-(1,0.1)	HC=10+10=20	HC=2+4=6	HC=1+3=4
Task 4: (0,0.3)-(0,1.5)	HC=10+10=20	HC=2+4=6	HC=1+3=4
Task 5: (1,0.5)-(0,1.2)	HC=10+10=20	HC=2+5=7	HC=1+4=5
Task 6: (1,0.7)-(0,1.5)	HC=8+8=16	HC=2+4=6	HC=1+3=4
Task 7: (0,1.0)-(1,0.0)	HC=12+12=24	HC=3+6=9	HC=2+5=7
Task 8: (0,0.8)-(1,1.6)	HC=8+8=16	HC=3+6=9	HC=2+5=7
Task 9: (0,0.7)-(0,1.1)	HC=2+2=4	HC=1+1+2=4	HC=1+1=2
Task 10: (1,1.5)-(1,0.2)	HC=8+8=16	HC=2+5=7	HC=1+4=5
Task 11: (0,1.3)-(0,0.1)	HC=10+10=20	HC=2+4=6	HC=1+3=4
Task 12: (0,1.4)-(1,0.6)	HC=8+8=16	HC=1+4=5	HC=0+3=3
Task 13: (1,0.1)-(1,1.1)	HC=6+6=12	HC=2+4=6	HC=1+3=4
Task 14: (1,1.2)-(0,0.8)	HC=8+8=16	HC=3+6=9	HC=2+5=7
Task 15: (1,0.6)-(0,1.2)	HC=4+4=8	HC=3+6=9	HC=2+5=7
Task 16: (1,0.4)-(0,1.7)	HC=14+14=28	HC=1+3=4	HC=0+2=2
Task 17: (1,1.3)-(0,1.3)	HC=6+6=12	HC=2+4=6	HC=1+3=4
Task 18: (0,1.2)-(1,1.0)	HC=2+2=4	HC=1+1+3=5	HC=1+1=2
Task 19: (0,0.1)-(1,0.7)	HC=10+10=20	HC=2+4=6	HC=1+3=4
Task 20: (1,0.2)-(0,1.6)	HC=20+20=40	HC=3+6=9	HC=2+5=7
Task 21: (0,0.6)-(0,0.5)	HC=6+6=12	HC=3+3=1=7	HC=3+3=6
Task 22: (1,0.7)-(1,0.8)	HC=2+2=4	HC=1+1+2=4	HC=1+1=2
Task 23: (0,1.4)-(0,1.2)	HC=4+4=8	HC=2+2+1=5	HC=2+2=4
Task 24: (1,1.6)-(1,1.2)	HC=8+8=16	HC=4+4+1=9	HC=4+4=8
Task 25: (0,1.7)-(0,1.1)	HC=4+4=8	HC=2+2+1=5	HC=2+2=4

Unlike WNoC-CD, WNoC-DDs has the advantage of skipping subnets for many cases as each subnet is accommodated with an individual directory. WNoC-DDs need not to wait for the

serving the requests of a source core from a subnet. But in WNoC-CD, the subnets request the centralized directory and the requests are queued and they must wait until their turn arise. Thus, the centralized directory adds an extra hop as well as delay due to waiting time in queue. The hop count in WNoC-CD needs more hops compared to WNoC-DDs model to update the directory to maintain data consistency for Tasks such as 9, 18, 21, 22, 23, 24, and, 25 as the network is designed with centralized directory. WNoC-DDs has the advantage of having wireless router with individual directory to each subnet and thus avoids extra hop counts compared to WNoC-CD and traditional mesh architecture.

4.5.3 Power Consumption

In calculating the power consumed for each task, number of cores, routers, and directories involved in reaching destination node from source node is identified. Then the power consumption for each task is calculated by using the assumptions in Table 4.1. WNoC-DDs consume less power when compared to other architectures as the directory in each subnet handles the data. The power consumption of each individual task can be observed in Table 4.9. The assumptions for calculating power is like proposed architecture 1. Due to the existence of directory in each subnet the power consumption from a subnet is high but the overall power consumption for a task is less as it avoids the subnet communication at every instant. For example, Task 1 power consumption can be easily inferred with the formula given below.

In WNoC-DDs:

Ptot = Psdd
Psdd=Pawrsn+(Pcwr*Ncwr)+Pddr core+ 3 (Pwl)=2.5+(3*1)+6 +3(1.1)= 14.8
Ptot=14.8

Ptot= Psdd + Pdsddr (For In-Subnet)
Pdsddr=Pawrsn+(Pcwr*Ncwr)+Pddr

Table 4.9: Power consumption compared to WNoC-DDs architecture

Different Scenarios	**Traditional Mesh (mW)**	**WNoC-CD (mW)**	**Proposed WNoC-DDs (mW)**
Task 1: (0,0.0)-(1,1.8)	P_1=24, P_2=24, P_3=25, P_{tot}=73	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2	P_{tot}=14.8
Task 2: (0,0.4)-(1,1.4)	P_1=24, P_2=24, P_3=25, P_{tot}=73	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2	P_{tot}=11.8
Task 3: (0,0.7)-(1,0.1)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 4: (0,0.3)-(0,1.5)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 5: (1,0.5)-(0,1.2)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 6: (1,0.7)-(0,1.5)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 7: (0,1.0)-(1,0.0)	P_1=27, P_2=27, P_3=25 P_{tot}=79	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=14.8
Task 8: (0,0.8)-(1,1.6)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=14.8
Task 9: (0,0.7)-(0,1.1)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 10: (1,1.5)-(1,0.2)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 11: (0,1.3)-(0,0.1)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 12: (0,1.4)-(1,0.6)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2	P_{tot}=11.8
Task 13: (1,0.1)-(1,1.1)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 14: (1,1.2)-(0,0.8)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=14.8
Task 15: (1,0.6)-(0,1.2)	P_1=11, P_2=11, P_3=25 P_{tot}=47	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=14.8
Task 16: (1,0.4)-(0,1.7)	P_1=31, P_2=31, P_3=25 P_{tot}=87	P_{sdr}=6.9, P_{cdr}=9.3 P_{tot}=16.2	P_{tot}=11.8
Task 17: (1,1.3)-(0,1.3)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 18: (0,1.2)-(1,1.0)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=11.8
Task 19: (0,0.1)-(1,0.7)	P_1=23, P_2=23, P_3=25 P_{tot}=71	P_{sdr}=9.9, P_{cdr}=9.3 P_{tot}=19.2	P_{tot}=11.8
Task 20: (1,0.2)-(0,1.6)	P_1=43, P_2=43, P_3=25 P_{tot}=111	P_{sdr}=12.9, P_{cdr}=9.3 P_{tot}=22.2	P_{tot}=14.8

Table 4.9 (continued)

Different Scenarios	Traditional Mesh (mW)	WNoC-CD (mW)	Proposed WNoC-DDs (mW)
Task 21: (0,0.6)-(0,0.5)	P_1=15, P_2=15, P_3=25 P_{tot}=55	P_{sdr}=15.9, P_{ds}=14.8 P_{tot}=30.7	P_{tot}= P_{sdd} + P_{dsddr} =20.8+17.5 =38.3
Task 22: (1,0.7)-(1,0.8)	P_1=7, P_2=7, P_3=25 P_{tot}=39	P_{sdr}=12.9, P_{ds}=8.5 P_{tot}=21.4	P_{tot}=17.8+8.5 =26.3
Task 23: (0,1.4)-(0,1.2)	P_1=11, P_2=11, P_3=25 Ptot=47	P_{sdr}=12.9, P_{ds}=11.8 P_{tot}=24.7	P_{tot}=17.8+14.5 =32.3
Task 24: (1,1.6)-(1,1.2)	P_1=19, P_2=19, P_3=25 P_{tot}=63	P_{sdr}=18.9, P_{ds}=17.8 P_{tot}=36.7	P_{tot}=23.8+20.5 =44.3
Task 25: (0,1.7)-(0,1.1)	P_1=11, P_2=11, P_3=25 P_{tot}=47	P_{sdr}=12.9, P_{ds}=11.8 P_{tot}=24.7	P_{tot}=17.8+14.5 =32.3

In WNoC-DDs architecture, the power consumed by each of the distributed directories with the wireless router (Pddr) is 6 units. In WNoC-DD, with the introduction of individual directory in every subnet, the update sync is easy and reduces hop count as well as traffic compared to WNoC-CD. The reduced hop count due to WNoC-DDs architecture should minimize the power consumption and offer better performance when compared with traditional mesh, and WNoC-CD architectures (see Tasks 1 to 20 in Table 4.9).

4.6 Simulation of Proposed Architecture 3

Unlike 36-core architectures of proposed 1 and 2, different workload is considered for the 64-core architectures with uniform and non-uniform partition of subnets. In this section, firstly 64-core architecture is evaluated with uniform partition where each subnet has equal number of cores but lacked in finding the exact center core. This is because the number of cores is even that is 16-core subnet in this architecture. If we investigate the subnet division, the possibility of becoming center core is equal to 4-core for example cores'-9, 10, 17, and 18 in subnet 0. The selection of any above cores as center core can ensure greater performance only to the tasks that are directly connected to it. The other neighbor cores in that subnet will not get enough benefit of the center

core and thus increases latency. Hence non-uniform subnets are introduced to overcome the latency issues. The thumb rule in determining the subnet size in non-uniform partition is to select odd number of cores like 9, 15, and 25. This sort of clustering, benefits to find the approximate center core and it brings a tradeoff superiority to all the neighbor cores in a subnet. Also, the small and large subnet division allows us to assign subnets for distinct application loads. As each subnet is assigned with a directory as well as wireless router like proposed architectures 1 and 2, the latency and power consumption can be reduced.

To reach an agreement, which partition is better, uniform or non-uniform, one should go through the examination of performance parameters. In these architectures, different jobs are considered as loads and the performance is observed as an average of all jobs as well as individual jobs. The performance parameters examined in these architectures are communication latency, hop count, and power consumption. In the proposed architecture 3, traditional mesh, traditional WNoC, and WNoC-CD discussions are avoided as it is significantly proven that distributed directory architecture performs better than the above-mentioned architectures.

4.6.1 Communication Latency

The communication latency is calculated on job basis and on individual task basis. When a request for data is processed through source core, the tool starts the initiating time and it counts on until the request is completed by obtaining the data from the destination core. In this architecture, we also include the latency in updating its directory. It is very essential to consider, to avoid data synchronizations. Like the other proposed architectures, adaptive XY routing algorithm is used and the performance is good even though the number of cores increased. The message or data between cores is in the form of packets. Every packet is not processed completely through intermediate cores.

The detailed calculations of communication latency can be observed in Table 4.10. There are 6 jobs in total and individual tasks are 31 in total.

Table 4.10: Communication latency of 64-core architecture with uniform and non-uniform subnets

Different Scenarios	Subtasks between Cores	Uniform Partition (ms)	Non-Uniform Partition (ms)
Job 1	18-54	3x4+4x4+4x4+40=84	0+2x4+0x4+40=48
	59-19	5x4+5x4+7x4+40=108	3x4+3x4+3x4+40=76
	19-51	4x4+5x4+6x4+40=100	2x4+3x4+2x4+40=68
	18-50	3x4+4x4+4x4+40=84	0+2x4+2x4+40=56
	58-26	4x4+5x4+6x4+40=100	2x4+3x4+2x4+40=68
Job 2	19-20	0+2x4+0x4+40=48	0+2x4+0x4+40=48
	60-51	4x4+5x4+6x4+40=100	4x4+2x4+3x4+40=76
	52-50	3x4+4x4+4x4+40=84	3x4+0+1x4+40=56
	24-20	4x4+4x4+5x4+40=92	4x4+3x4+4x4+40=84
Job 3	6-28	3x4+4x4+4x4+40=84	3x4+5x4+5x4+40=92
	31-20	5x4+3x4+3x4+40=84	3x4+4x4+4x4+40=84
	63-39	5x4+4x4+2x4+40=80	3x4+5x4+5x4+40=92
	59-35	5x4+4x4+2x4+40=80	3x4+5x4+5x4+40=92
Job 4	19-35	4x4+5x4+6x4+40=100	2x4+4x4+1x4+40=68
	17-34	2x4+4x4+3x4+40=76	2x4+3x4+2x4+40=68
	38-14	3x4+3x4+3x4+40=76	3x4+2x4+2x4+40=68
	49-52	2x4+4x4+3x4+40=76	2x4+3x4+2x4+40=68
	23-20	4x4+3x4+2x4+40=76	2x4+4x4+3x4+40=76
	4-31	3x4+5x4+5x4+40=92	5x4+4x4+5x4+40=96
	60-39	4x4+4x4+5x4+40=92	4x4+5x4+6x4+40=100
Job 5	54-63	3x4+5x4+1x4+40=76	0+3x4+1x4+40=56
	53-55	2x4+4x4+1x4+40=68	2x4+2x4+1x4+40=60
	47-61	3x4+4x4+3x4+40=76	3x4+3x4+3x4+40=76
	63-62	0+2x4+0x4+40=48	0+2x4+0+40=48
	53-45	2x4+0x4+0x4+40=48	2x4+0+0+40=48
	46-62	2x4+4x4+1x4+40=68	2x4+2x4+1x4+40=60
	55-47	0+2x4+0x4+40=48	0+2x4+0+40=48
Job 6	9-45	0+2x4+0x4+40=48	3x4+4x4+4x4+40=84
	54-50	3x4+4x4+4x4+40=84	0+2x4+0+40=48
	18-22	3x4+4x4+4x4+40=84	0+2x4+0+40=48
	13-41	0+2x4+0x4+40=48	3x4+4x4+4x4+40=84

The critical combination of a packet are header, payload, and trailer. The cores just forward the packets to its neighbors if they are not the destination core, which is revealed from the header

packet. The latency between intermediate cores through a single hop is four units and the destination core include a latency of 40 units as it must read the full packet. These assumptions are like previous architectures and the details are well explained in the above sections for 36-core architectures. An individual core always knows, its one hop distance cores that is East, West, North, and South cores. For any task, the latency is evaluated by considering the number of cores involved in the process from source to destination and vice-versa. Generally, the cores involved in each task is based on the routing strategy and size of subnet where partition plays a vital role. On average, non-uniform partition has more potential than uniform and the results are quite satisfactory.

The path calculation for computing latency in mathematical representation is as follows:

Latency= Know the destination (Source to Directory) + Directory request to destination core (Directory to destination) + Send data from destination to source core directly.

By using the above mathematical expression, the latency for each job is calculated for both uniform and non-uniform partitions and then finally compared to analyze the performance.

4.6.2 Hop Count

Hop is a link between two cores that is wired, or wireless connected. Cores communicate each other with these links and the error-free connection ensures trustworthy communication. The computation of hop count is analyzed for job and individual task basis. The hop count in both uniform and non-uniform partitions, does not require any return or acknowledgement. This approach will reduce the number of hops in larger when compared to traditional mesh architectures. Number of hops required for any job is based on the number of links that are connected to cores. If a greater number of hops involved for data transmission, then the comprehensive hop counts of that task will be larger.

The detailed calculations of latency can be observed in Table 4.11. There are six jobs in total and 31 individual tasks in total.

Table 4.11: Hop count of 64-core architecture with uniform and non-uniform subnets

Different Scenarios	Subtasks between Cores	Uniform Partition	Non-Uniform Partition
Job 1	18-54	2+3+5=10	0+1+1=2
	59-19	4+4+8=16	2+2+4=8
	19-51	3+4+7=14	1+2+3=6
	18-50	2+3+5=10	0+1+1=2
	58-26	3+4+7=14	1+2+3=6
Job 2	19-20	0+1+1=2	1+1+1=3
	60-51	3+4+7=14	3+1+2=6
	52-50	2+3+5=10	2+0+2=4
	24-20	3+3+6=12	3+2+5=10
Job 3	6-28	2+3+5=10	2+4+6=12
	31-20	4+2+4=10	2+3+5=10
	63-39	4+3+3=10	2+4+6=12
	59-35	4+3+3=10	2+4+6=12
Job 4	19-35	3+4+7=14	1+3+2=6
	17-34	1+3+4=8	1+2+3=6
	38-14	2+2+4=8	2+1+3=6
	49-52	1+3+4=8	1+2+3=6
	23-20	3+2+3=8	1+3+4=8
	4-31	2+4+6=12	4+3+7=14
	60-39	3+3+6=12	3+4+7=14
Job 5	54-63	2+4+2=8	0+2+2=4
	53-55	1+3+2=6	1+1+2=4
	47-61	2+2+4=8	2+2+4=8
	63-62	1+1+4=6	1+1+1=3
	53-45	1+0+1=2	0+1+1=2
	46-62	1+3+2=6	1+1+2=4
	55-47	1+1+4=6	1+1+1=3
Job 6	9-45	0+1+1=2	2+3+5=10
	54-50	2+3+5=10	0+1+1=2
	18-22	2+3+5=10	0+1+1=2
	13-41	0+1+1=2	2+3+5=10

In these architectures, directories play a key role as they define the path between source and destination cores. However, selection of center core is complicated and so partition of subnets

are given highest priority which will resolve the performance of the system. Unlike uniform partition, non-uniform partition has small and large subnets. Eventually, smaller subnets require less hops when compared to larger subnets. However, it takes more hops if larger subnet is considered. As the workload is random, the outcome performance in both partition methods will determine the best approach of logical splitting of subnets.

Two cores communicate directly only when they are at one hop distance. However, after data exchange between cores, it is essential to update the directory to get rid of data synchronization issues. In such cases, mesh topology may be advantage but on average of random workloads, the directory-based architectures proved their performance is immense. We can also write an algorithm to check the number of hops required for any task in advance before transmitting data and we can decide to follow mesh topology path or directory path. But this approach typically increases the delay and power consumption as they must compute multiple paths and logics possible. So, directory-based architecture with non-uniform partition assures a trade-off solution for large core architectures.

The calculation of computing total hops involved in data transmission can be expressed in mathematical representation as follows:

Hop Count= Know the destination (Source to Directory) + Directory request to destination core (Directory to destination) + Send data from destination to source core directly.

By using the above mathematical expression, the hop count for each job is calculated on both uniform and non-uniform partitions and then finally compared to analyze the performance.

4.6.3 Power Consumption

Power consumption is one of the major parameters, where this is highly concerned to consider any architecture. This is a bit complex to compute compared to latency and hop count

calculations. The total power consumption includes the involvement of cores, routers, directories, subnets, etc. As the directories are superintendent or controller of the individual subnets, it is assumed that the power consumption of directory is six units, which is double to an ordinary core. For data transmission, if the number of cores involved is less eventually, it reduces the power consumption in total. The detailed calculations of power consumption can be observed in Table 4.12. There are six jobs in total and individual tasks are 31 in total.

Table 4.12: Power consumption of 64-core architecture with uniform and non-uniform subnets

Different Scenarios	**Subtasks between Cores**	**Uniform Partition (mW)**	**Non-Uniform Partition (mW)**
Job 1	18-54	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6
	59-19	P_{sd} =36.3, P_{ds}=36.3 P_{tot}=72.6	P_{sd} =24.3, P_{ds}=24.3 P_{tot}=48.6
	19-51	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6	P_{sd} =21.3, P_{ds}=21.3 P_{tot}=42.6
	18-50	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6
	58-26	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6	P_{sd} =21.3, P_{ds}=21.3 P_{tot}=42.6
Job 2	19-20	P_{sd} =18, P_{ds}=18.3 P_{tot}=36.3	P_{sd} =6, P_{ds}=18.3 P_{tot}=24.3
	60-51	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6	P_{sd} =24, P_{ds}=12.3 P_{tot}=36.3
	52-50	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=12 P_{tot}=27.3
	24-20	P_{sd} =30.3, P_{ds}=30.3 P_{tot}=60.6	P_{sd} =30.3, P_{ds}=18 P_{tot}=48.3
Job 3	6-28	P_{sd} =30.3, P_{ds}=18 P_{tot}=48.3	P_{sd} =30.3, P_{ds}=30.3 P_{tot}=60.6
	31-20	P_{sd} =33.3, P_{ds}=15 P_{tot}=48.3	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6
	63-39	P_{sd} =36.3, P_{ds}=12 P_{tot}=48.3	P_{sd} =30.3, P_{ds}=30.3 P_{tot}=60.6
	59-35	P_{sd} =36.3, P_{ds}=12 P_{tot}=48.3	P_{sd} =30.3, P_{ds}=30.3 P_{tot}=60.6

Table 4.12 (continued)

Different Scenarios	Subtasks between Cores	Uniform Partition (mW)	Non-Uniform Partition (mW)
Job 4	19-35	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6	P_{sd} =27.3, P_{ds}=9 P_{tot}=36.3
	17-34	P_{sd} =24.3, P_{ds}=24.3 P_{tot}=48.6	P_{sd} =24.3, P_{ds}=12 P_{tot}=36.3
	38-14	P_{sd} =24.3, P_{ds}=24.3 P_{tot}=48.6	P_{sd} =24.3, P_{ds}=15 P_{tot}=39.3
	49-52	P_{sd} =24.3, P_{ds}=24.3 P_{tot}=48.6	P_{sd} =24.3, P_{ds}=15 P_{tot}=39.3
	23-20	P_{sd} =30.3, P_{ds}=12 P_{tot}=42.3	P_{sd} =24.3, P_{ds}=24.3 P_{tot}=48.6
	4-31	P_{sd} =33.3, P_{ds}=21 P_{tot}=54.3	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6
	60-39	P_{sd} =33.3, P_{ds}=21 P_{tot}=54.3	P_{sd} =33.3, P_{ds}=33.3 P_{tot}=66.6
Job 5	54-63	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6
	53-55	P_{sd} =27.3, P_{ds}=9 P_{tot}=36.3	P_{sd} =21.3, P_{ds}=12 P_{tot}=33.3
	47-61	P_{sd} =27.3, P_{ds}=15 P_{tot}=42.3	P_{sd} =27.3, P_{ds}=15 P_{tot}=42.3
	63-62	P_{sd} =27.3, P_{ds}=6 P_{tot}=33.3	P_{sd} =21.3, P_{ds}=6 P_{tot}=27.3
	53-45	P_{sd} =12.3, P_{ds}=9 P_{tot}=21.3	P_{sd} =18.3, P_{ds}=6 P_{tot}=24.3
	46-62	P_{sd} =27.3, P_{ds}=9 P_{tot}=36.3	P_{sd} =15.3, P_{ds}=12 P_{tot}=27.3
	55-47	P_{sd} =24.3, P_{ds}=6 P_{tot}=30.3	P_{sd} =18.3, P_{ds}=6 P_{tot}=24.3
Job 6	9-45	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6
	54-50	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6
	18-22	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6
	13-41	P_{sd} =15.3, P_{ds}=15.3 P_{tot}=30.6	P_{sd} =27.3, P_{ds}=27.3 P_{tot}=54.6

As discussed earlier, size of subnets and partitions also plays a critical role in computing power consumption. Total power consumption includes two paths, where the power consumed in

source to destination path as request for data and return path that is from destination to source as accomplishment path.

There are different scenarios of workload and the involvement of directories as controllers and the ability of ordinary cores with single hop connected neighbors will have different routing methods. Thus, the values or amount of power consumption differs according to task wise. The principle of computing power in both uniform and non-uniform partition is identical, however the values differ due to the number of directories, routers, cores, and subnets involved.

The calculation of computing power consumption is represented in mathematical form and it is described as follows:

Power consumption= Power consumed from source to destination (P_{sd}) + Power consumed from source to destination (P_{ds}).

For Out-subnet:

$P_{sd} = (P_{cwr} \times N_{cwr}) + 2\ P_{ddr} + 3\ (P_{wl})$

$P_{ds} = (P_{cwr} \times N_{cwr}) + 2\ P_{ddr} + 3\ (P_{wl})$

For Out-subnet: (Directory-Directory)

$P_{sd} = 2\ P_{ddr} + 3\ (P_{wl})$

$P_{ds} = 2\ P_{ddr} + 3\ (P_{wl})$

For Out-subnet/In-subnet: (One hop)

When the data exchange is within one hop, they communicate directly and simply update directory, which is beneficiary in not requesting the directory that may increase power consumption.

$P_{sd} = (P_{cwr} \times N_{cwr}) + 2\ P_{ddr}$

$P_{ds} = (P_{cwr} \times N_{cwr})$ + Power from source to directory intermediate cores (P_{sdr})+3 (P_{wl})

For In-subnet:

The data transmission for in-subnet is slightly different from other scenarios. If they are not at one hop distance, then the request goes to directory, but the return path is not necessarily through directory as they are in same subnet. The directory informs the destination core to send the data directly to source if the directory finds the destination is not busy.

P_{sd} = Power from source to directory (P_{sdr}) + Power from directory to destination

P_{ds} = Power from destination to source + Update other directories

According to the experimental results, the power consumption in non-uniform subnets may be higher for some special tasks (e.g., Job 6 Subtasks between cores 9-45 and 13-41), but on average, the performance of non-uniform subnets compared to uniform partition of subnets with large number of cores is impressive.

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, first we discuss results of the proposed WNoC-CD, and WNoC-DDs architectures. Then, we discuss the results of the non-uniform partition of subnets with WNoC-DDs as proposed architecture 3. For proposed architectures 1 and 2, the same workload is used. We use 25 different communication tasks as workload by considering in-subnet and out-subnet scenarios. Then the performance characteristics such as communication latency, hop count, and power consumption are derived for both WNoC-CD, and WNoC-DDs architectures. For proposed architecture 3, distributed directories with uniform and non-uniform partition is the major consideration and they are evaluated using six different jobs which are subdivided into 31 individual sub tasks. The results of proposed architectures 1 and 2 are discussed in the following subsections.

5.1 Evaluation of Proposed Architecture 1

In this work, we introduce centralized directory with WNoC architecture and is discussed as proposed architecture 1. The performance of each architecture is clearly observed when the comparison is performed according to each task.

5.1.1 Communication Latency

In this work, we considered 20 scenarios for out-subnet and 5 scenarios for in-subnet. The latency is same for all the in-subnet tasks as they are basically the mesh architecture. The path for in-subnet tasks are identical. The hop count and power consumption may be different, because of the wireless routers and directory. Figure 5.1 illustrates the communication latency due to the mesh, traditional WNoC, and proposed WNoC-CD architectures for all 25 tasks.

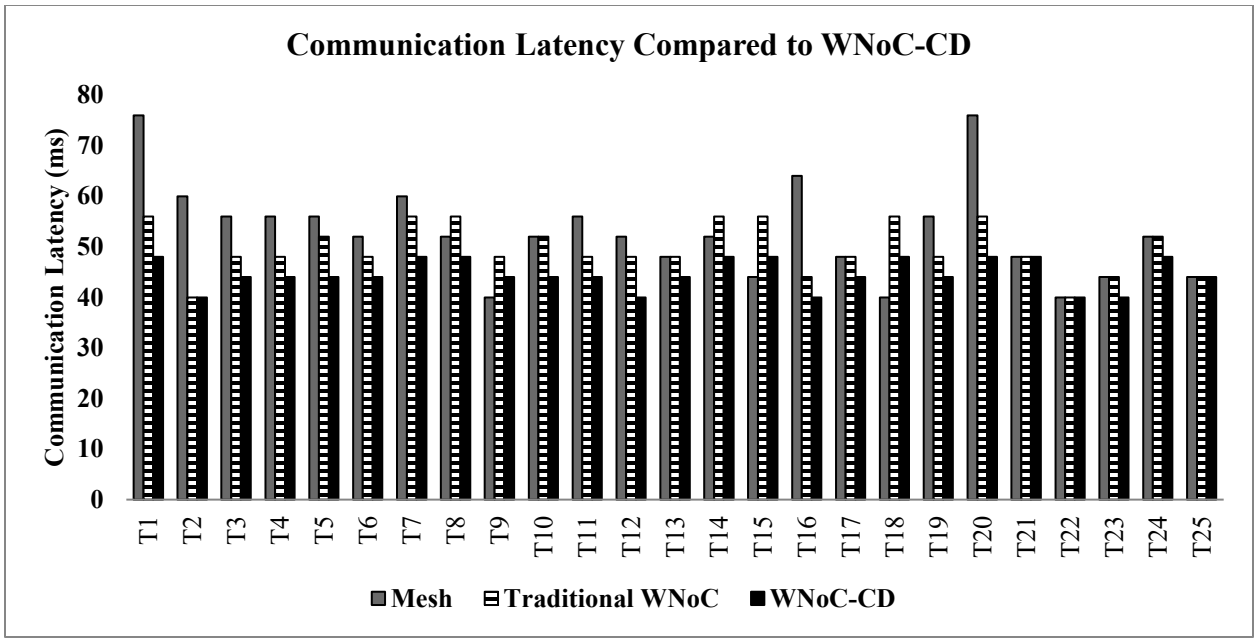


Figure 5.1: Communication latency compared to WNoC-CD architecture

From Figure 5.2, by considering all the tasks, it is observed that the WNoC-CD architecture help reduce the communication latency, in an average, by 16.01% compared to mesh architecture, and 10.32% compared to traditional WNoC architecture.

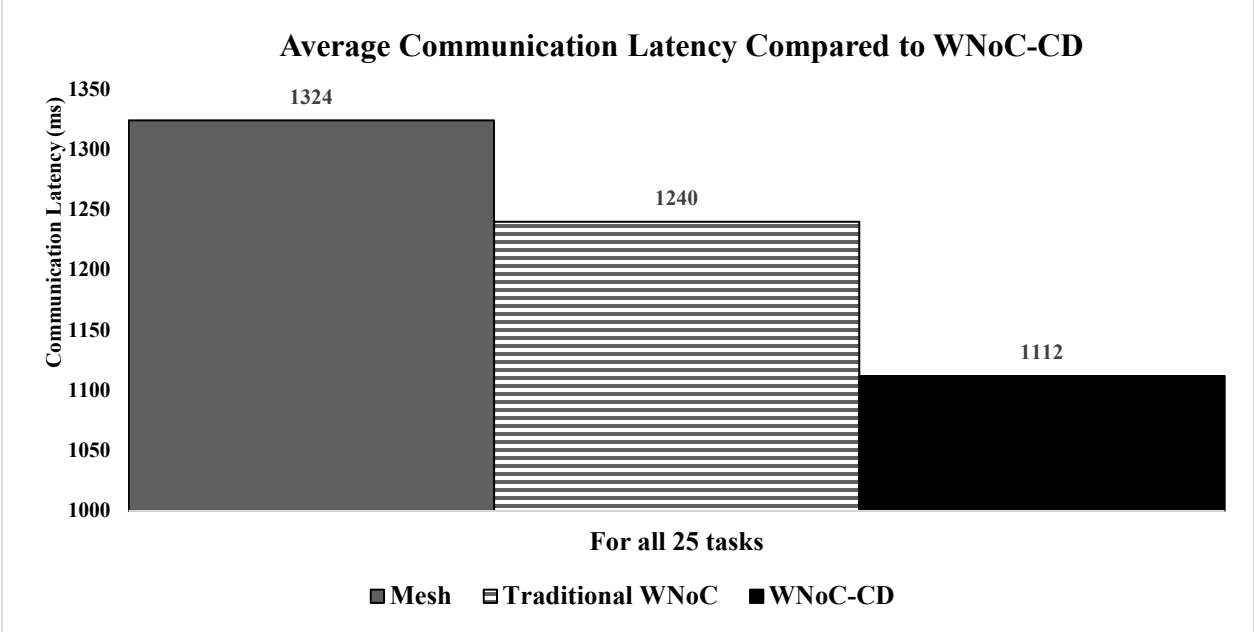


Figure 5.2: Average communication latency compared to WNoC-CD architecture

5.1.2 Hop Count

The hop counts due to the mesh, WNoC, and proposed WNoC-CD architectures for all 25 tasks are illustrated in Figure 5.3. WNoC-CD reduces extra hops compared to mesh or traditional broadcasting WNoC as the directory supervises the routers in establishing the path between source and destination cores. However, WNoC requires few extra hops based on the task to update the directory for maintaining data sync. This process does not affect the performance compared to mesh and traditional WNoC architectures. In traditional WNoC, for Task 14 and Task 15, the hop count is maximum when compared to other architectures. For Task 15 in traditional WNoC, even though the distance is between the source and destination is two hops, as they were in two different subnets, the path is through wireless routers which makes long path compared all other architectures. To process any request irrespective of destination subnet, the directory handles the request and ensures delivery. If the destination core is at one hop distance, then the data transfer takes place directly and updates the directory which is considered as extra hops compared to mesh architecture.

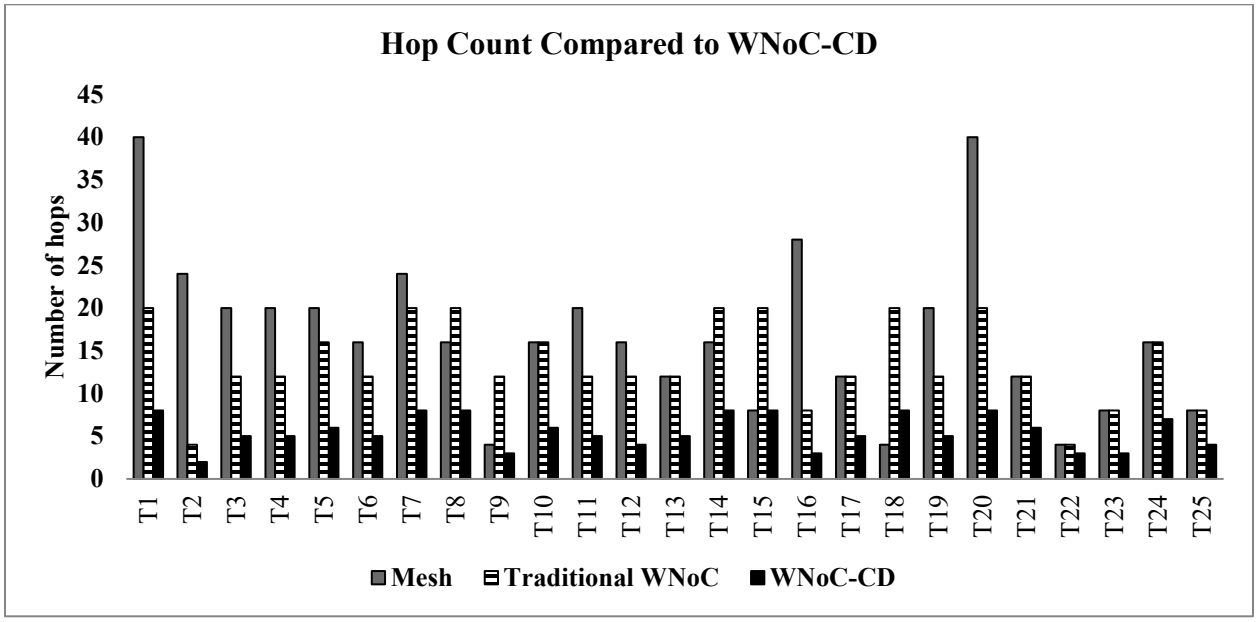


Figure 5.3: Hop count compared to WNoC-CD architecture

From Figure 5.4, it can be observed that the hop count due to the WNoC-CD architecture is reduced by 62.03% when compared with the mesh architecture, and 52.65% when compared to traditional WNoC architecture.

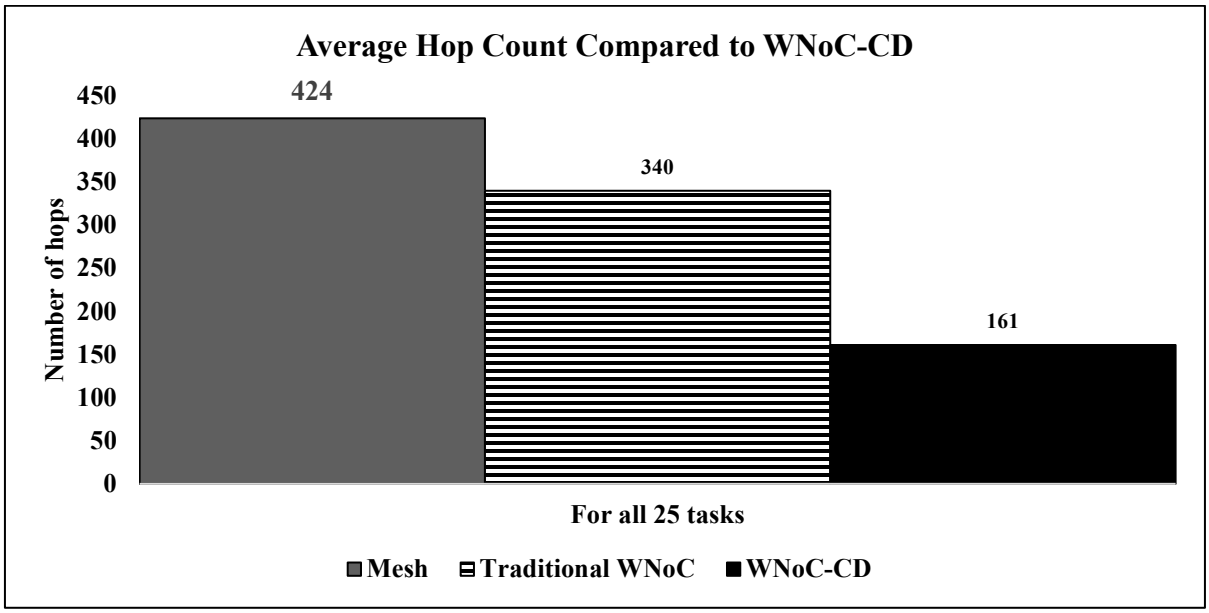


Figure 5.4: Average hop count compared to WNoC-CD architecture

5.1.3 Power Consumption

Power consumption due to the mesh, traditional WNoC, and proposed WNoC-CD architectures for all the tasks is illustrated in Figure 5.5. It should be noted that the amount of power consumption is different in each subnet as broadcasting is involved in traditional WNoC and the update of data is essential with the centralized directory. Thus, the power consumption varies for each individual task. The power consumption in mesh architecture is maximum because the average of the whole network is considered, whereas the traditional WNoC and WNoC-CD architectures consumes less power due to the introduction of subnets. WNoC-CD architecture consumes power based on subnet usage and the non-active subnets are idle and the power consumption of idle subnets is negligible.

From Figure 5.6, it can be observed that the power consumption due to the proposed WNoC-CD architecture is reduced by 66.96% when compared with that of the mesh architecture, and 57.3% when compared with traditional WNoC architecture.

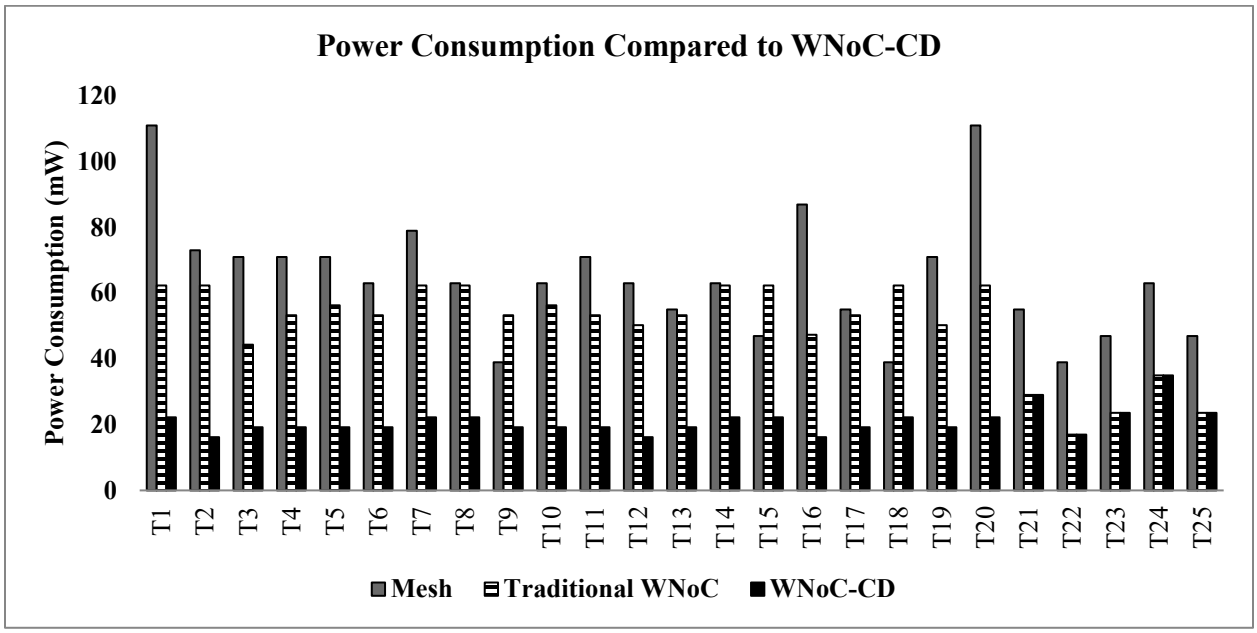


Figure 5.5: Power consumption compared to WNoC-CD architecture

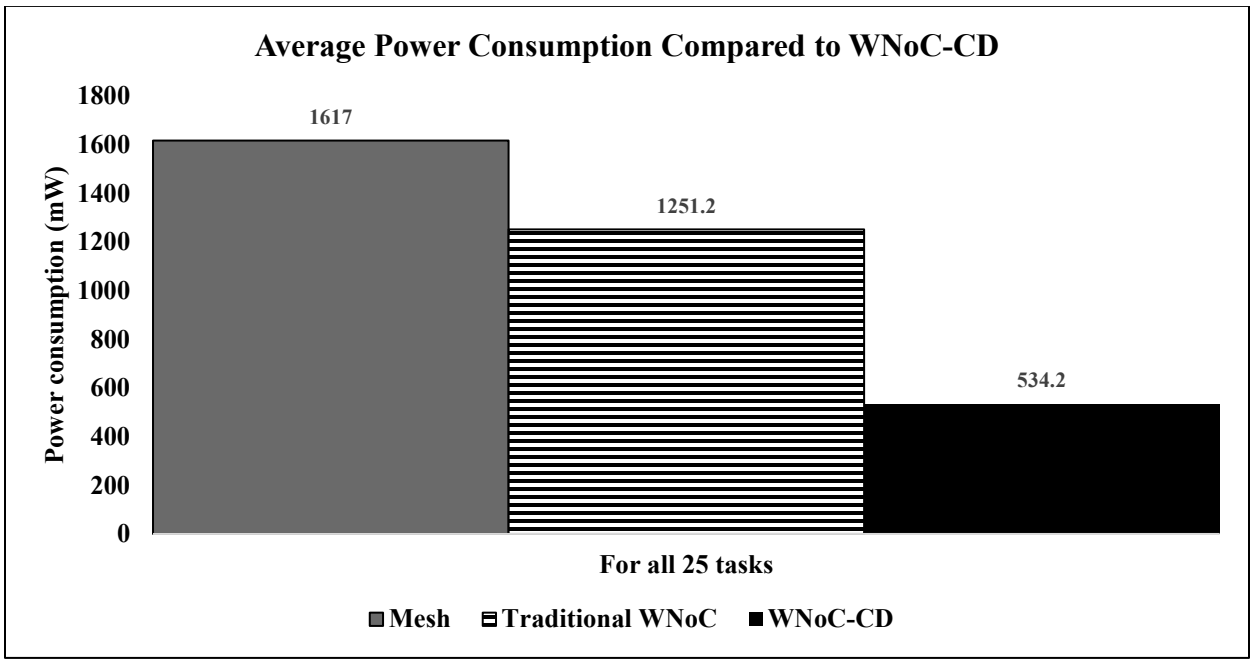


Figure 5.6: Average power consumption compared to WNoC-CD architecture

5.2 Evaluation of Proposed Architecture 2

In this work, we introduce directory in each subnet and so it is termed as WNoC architecture with distributed directories (WNoC-DDs). The performance of each architecture is clearly observed when the comparison is performed according to each task. WNoC-DDs architecture results are promising in reducing the latency as well as hop counts and thus reduces power consumption. This performance achievement is possible with the directory in each subnet. The directory knows the address of each subnet and the core as well as the directory associated to the subnet. Particularly, this capability of directory makes the communication between cores more flexible and reduces the pressure of data synchronization at core level. Clustering subnets and communicating through subnets is like divide and conquer strategy, which enhances the performance of the architecture.

5.2.1 Communication Latency

In this work, we considered 25 tasks that has 20 scenarios for out-subnet and 5 scenarios for in-subnet. If we observe the latency for in-subnet scenarios, it is clear they are performing identical and is obvious as they are basically mesh architecture. However, with the introduction of directory in each subnet, the hop count and power consumption may be different for proposed WNoC-DDs architecture. Figure 5.7 illustrates the communication delay due to the mesh, WNoC-CD, and proposed architectures for all 25 tasks.

For all tasks, the latency due to WNoC-DDs architecture is smaller or same compared to that due to the mesh and WNoC-CD architectures. For Tasks 9, 18, and 22, the cores are next to each other; WNoC-DDs, mesh and WNoC-CD provides the least and same latency as they relate to one hop distance.

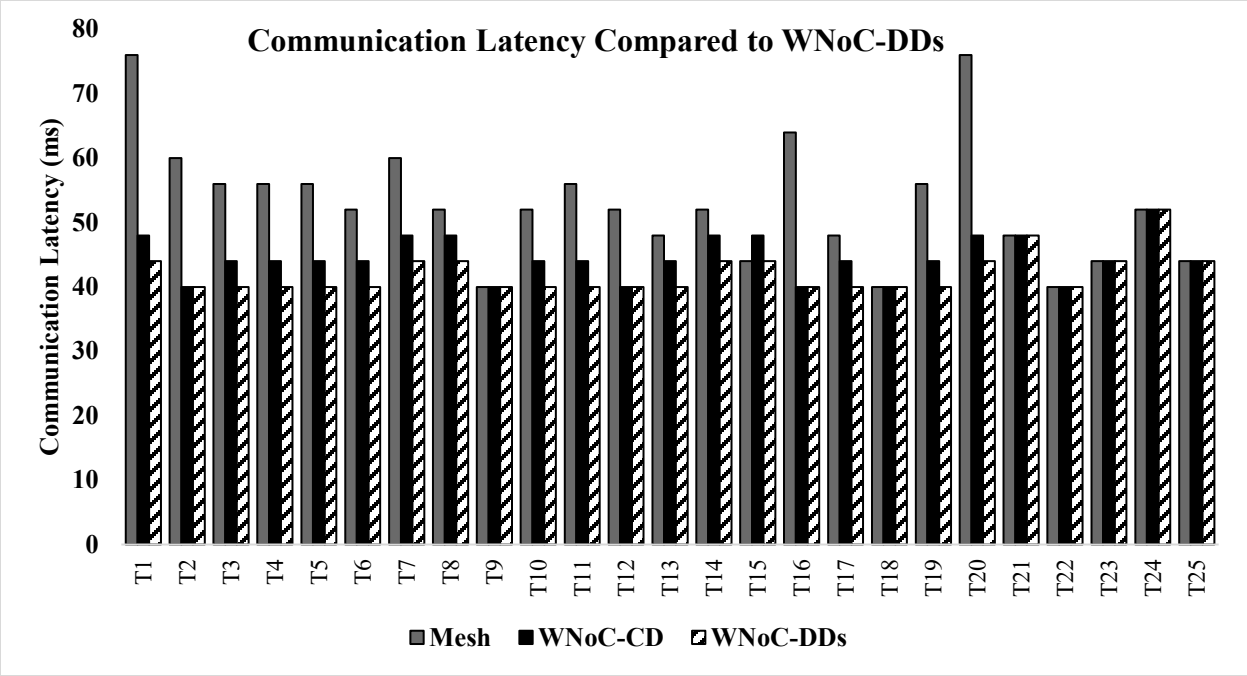


Figure 5.7: Communication latency compared to WNoC-DDs architecture

From Figure 5.8, by considering all the tasks, it is observed that the WNoC-DDs architecture help reduce the communication delay, in an average, by 20.54% compared to mesh architecture, and 5.40% compared to WNoC-CD architecture.

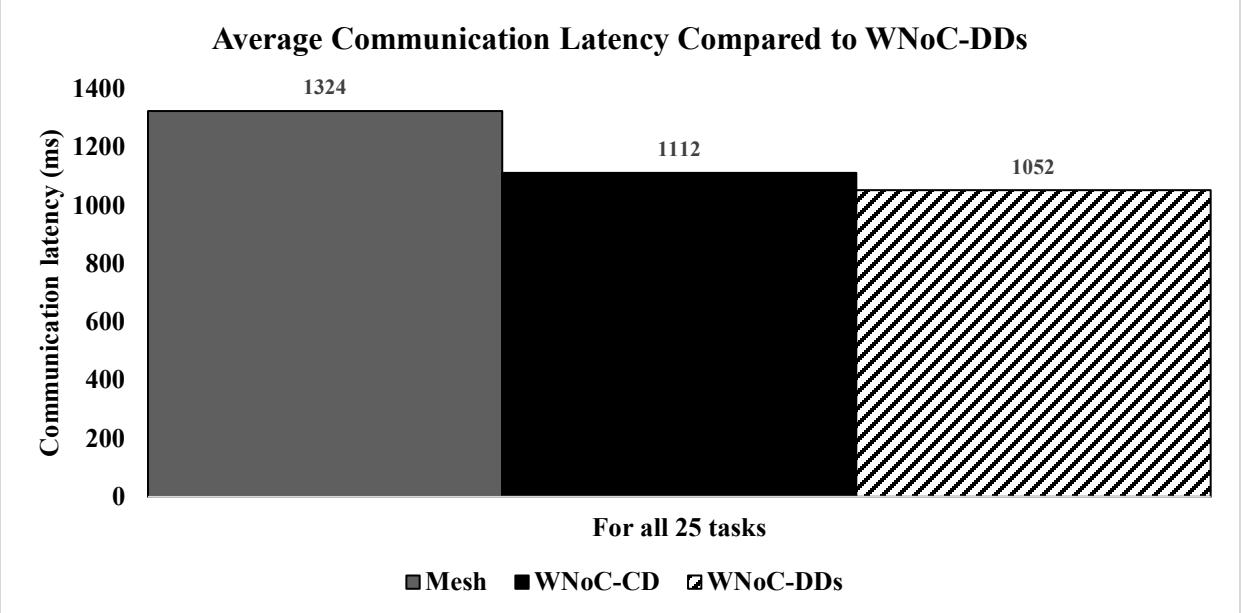


Figure 5.8: Average communication latency compared to WNoC-DDs architecture

5.2.2 Hop Count

The hop counts due to the mesh, WNoC-CD, and proposed WNoC-DDs architectures for all 25 tasks are illustrated in Figure 5.9. WNoC-CD needs extra hops (based on task) to update the centralized directory for maintaining data sync. In WNoC, for Task 14 and Task 15, the hop count is maximum when compared to other architectures. The directory handles the request of each source core and ensures data transfer from the destination core to source core. WNoC-DDs architecture broadcasts the accomplished task information to other subnets for maintaining the data sync among directories/subnets.

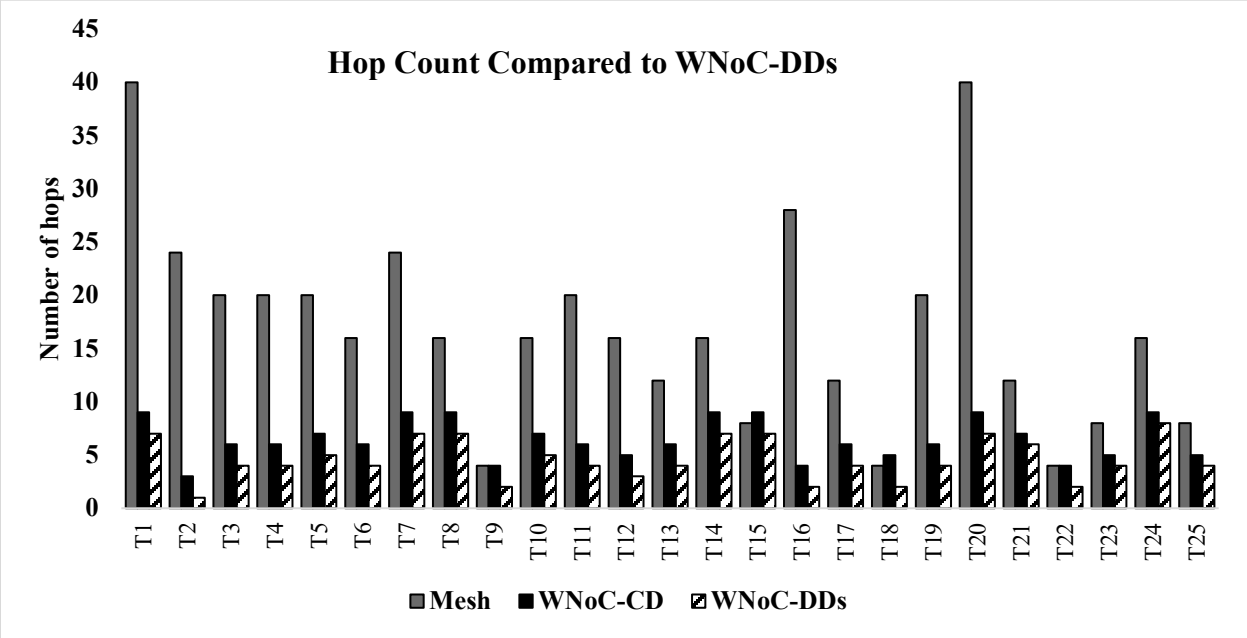


Figure 5.9: Hop count compared to WNoC-DDs architecture

From Figure 5.10, it can be observed that the hop count due to the WNoC-DDs architecture is reduced by 73.11% when compared with the mesh architecture, and 29.19% when compared to WNoC-CD architecture. This is because the WNoC-DDs architecture does not require any acknowledgement or any return path to the source to complete the task.

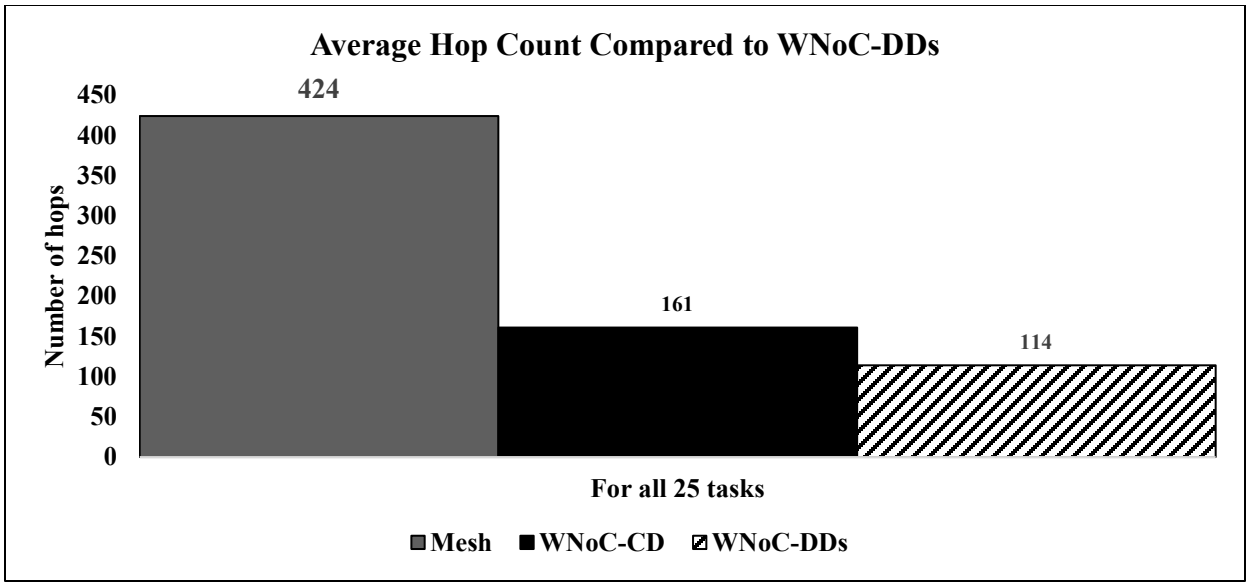


Figure 5.10: Average hop count compared to WNoC-DDs architecture

5.2.3 Power Consumption

Power consumption due to the mesh, WNoC-CD, and proposed WNoC-DDs architectures for all the tasks is illustrated in Figure 5.11. It should be noted that the amount of power consumption should be increased within the subnet, as the distributed directories with wireless router is present in each subnet.

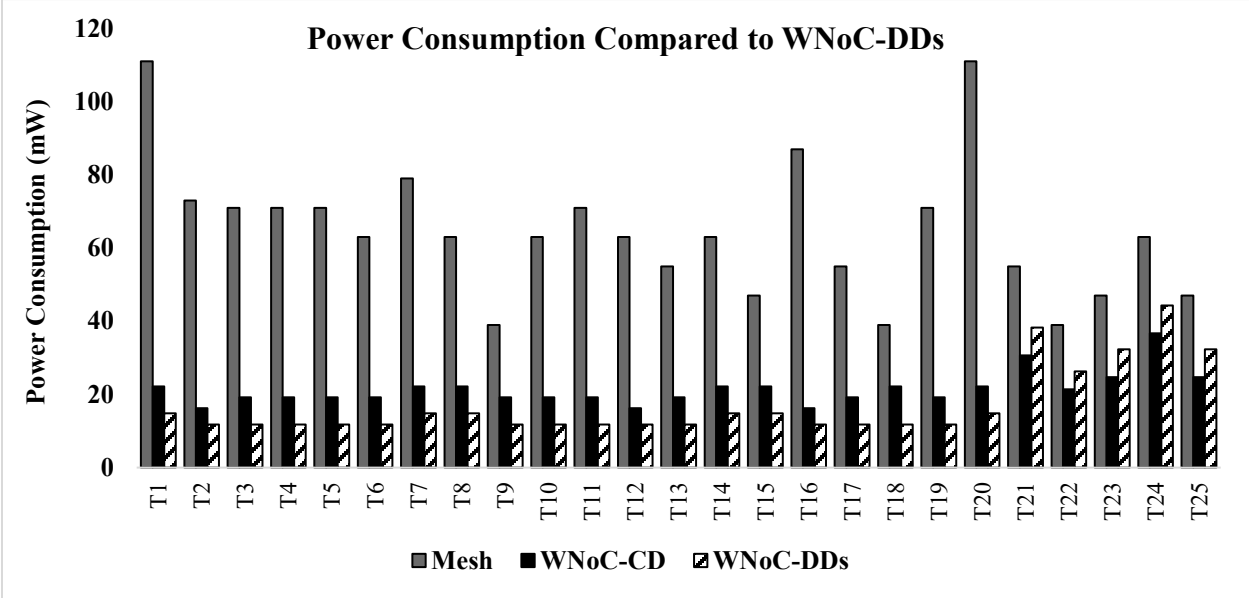


Figure 5.11: Power consumption compared to WNoC-DDs architecture

For Tasks 21, 22, 23, 24, and 25 the power consumption is slightly large compared to WNoC-CD, but it is suitable for large network with reduced traffic. However, the proposed architecture power consumption is less for any individual task when compared to mesh architecture.

From Figure 5.12, it can be observed that the power consumption due to the proposed architecture is reduced by 73.56% when compared with that of the mesh architecture, and 19.97% when compared with that of the WNoC-CD architecture. This is because the proposed distributed directories (Pddr) take less power than a centralized directory (Pcdr) with wireless core/router.

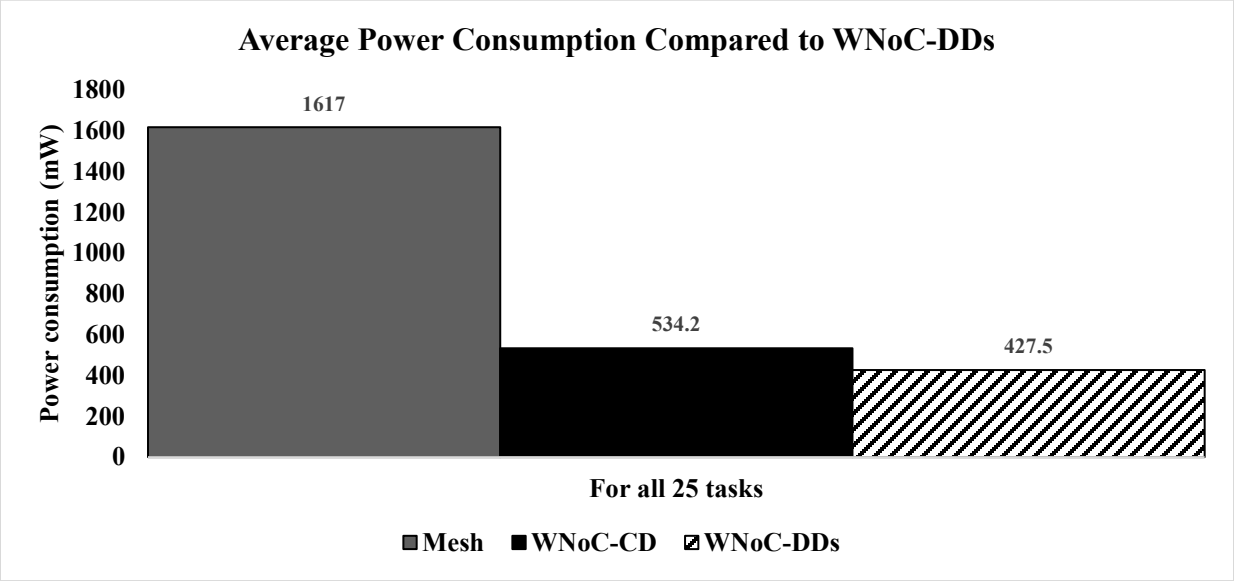


Figure 5.12: Average power consumption compared to WNoC-DDs architecture

5.3 Evaluation of Proposed Architecture 3

In this work, we considered six jobs where each job has individual subtasks and there are 31 tasks in total. The performance of these architectures is analyzed based on job basis and on average of all jobs. Even though both uniform and non-uniform partitions are based on WNoC-DDs architecture, the performance differs due to the shift of center cores and the size of subnets. The results proved that with proper interpretation of subnets clustering, can bring significant

performance of the architecture. In this section, we compare the performance parameters such as latency, hop count, and power consumption in both uniform and no-uniform partition of subnets.

5.3.1 Communication Latency

In this work, we considered 6 jobs where each job has individual subtasks. The performance of these architectures is analyzed based on job basis. However, the individual job performance could affect based on the individual tasks in number and short and large distance between source and destination. The latency for uniform and non-uniform subnets for 64-core architecture is illustrated in Figure 5.13.

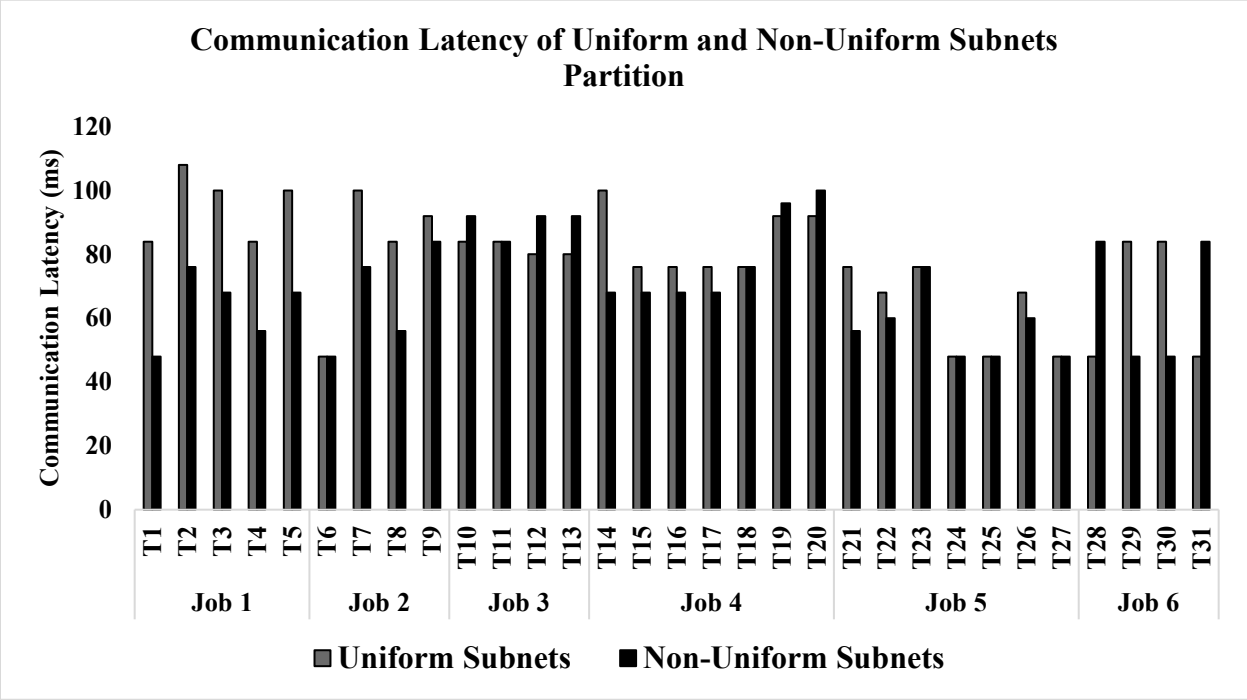


Figure 5.13: Communication latency of uniform and non-uniform subnets in 64-core architecture

In these architectures, latency is determined from end-to-end message for data request and data fetching. The latency is minimum if the cores are directly connected and it is maximum for first core to last core. For any task, it is essential to update the directory and so it can maintain data synchronization. Few in-subnet jobs may take more time and it varies based on the size of the subnet. In uniform subnets, mostly the distance is identical and so the performance is steady. With

the non-uniform partitions, where small to large subnets existence can improve performance for many jobs on random. On average, non-uniform subnets perform well when compared to uniform.

Figure. 5.14 illustrates the average performance according to the job. From the Figure 5.14, it can be observed that non-uniform partition of subnets performs better in 4 jobs out of 6 jobs. Job 6 performance is identical as the jobs are from directory to directory. These random workloads generated by VisualSim tool provides substantial information that non-uniform partition of subnets perform well.

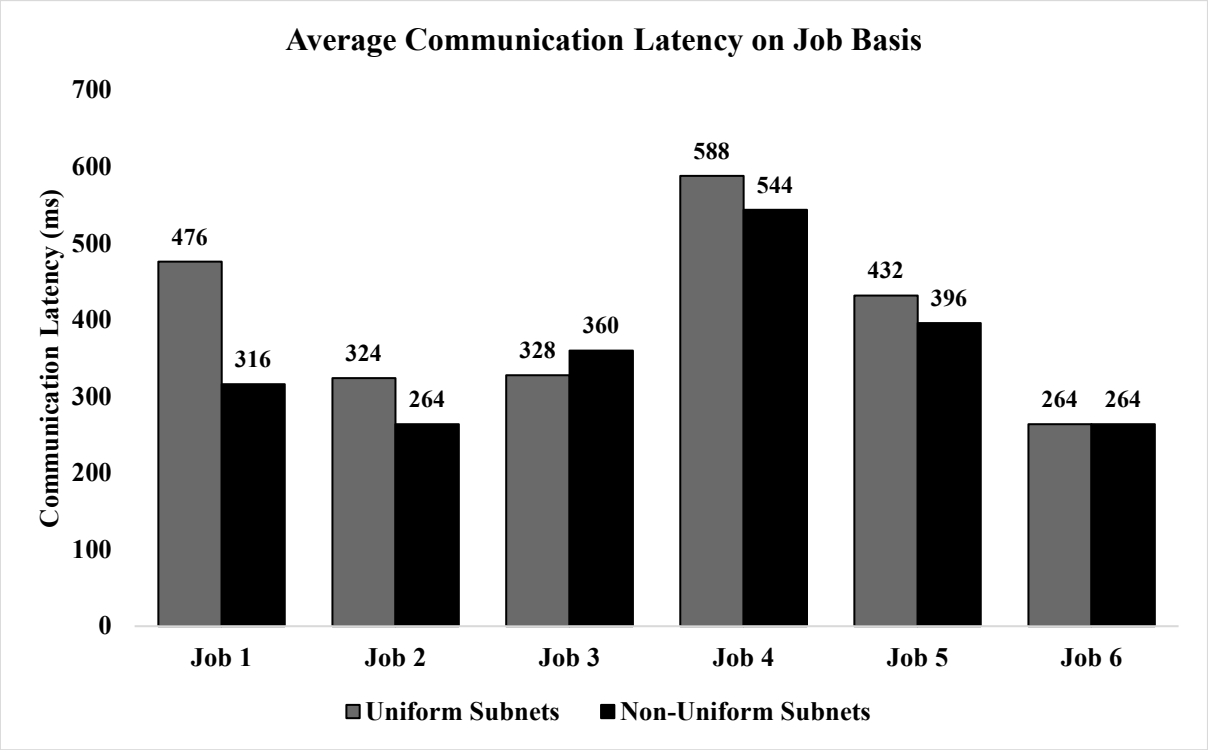


Figure 5.14: Average communication latency on job basis

From Figure 5.15, by considering all the tasks, it is observed that the non-uniform subnets 64-core architecture help reduce the communication delay, in an average, by 11.11% compared to uniform subnet architecture.

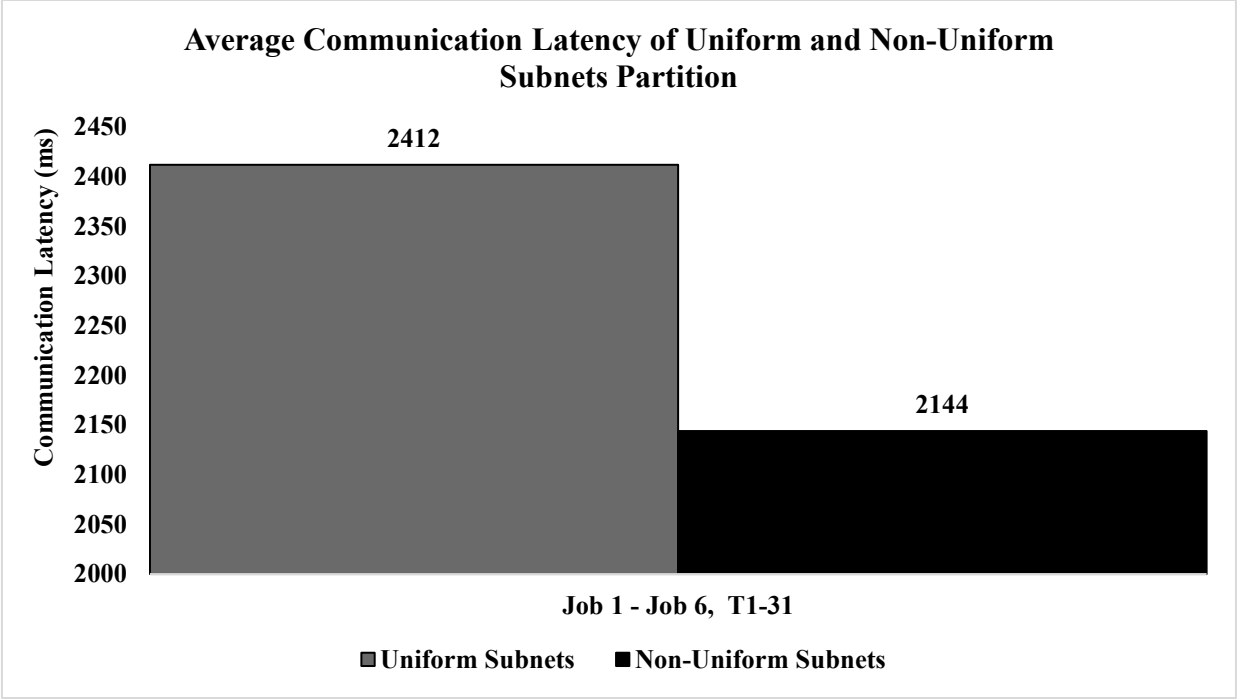


Figure 5.15: Average communication latency of 64-core architecture

5.3.2 Hop Count

The hop counts for uniform and non-uniform subnets architectures for all 6 jobs that has 31 individual subtasks are illustrated in Figure 5.16. In these architectures, the basic working principle is identical where each subnet has a directory and wireless router. The working principle of the architectures is like WNoC-DDs 36-core architecture. However, few advancements applied in the algorithm to select the paths and considered the forward and return paths of communication. So, the description is mostly by considering the partition strategy. The modifications in algorithm agreed more practical approach. The workload on job basis for uniform and non-uniform architectures, notifies their capacity and range. The drawbacks can be monitored closely, and it could help to change logical modifications if possible.

Figure 5.17 illustrates the average performance of hop count according to the job. Like the latency, the hop count performs well with non-uniform partition of subnets.

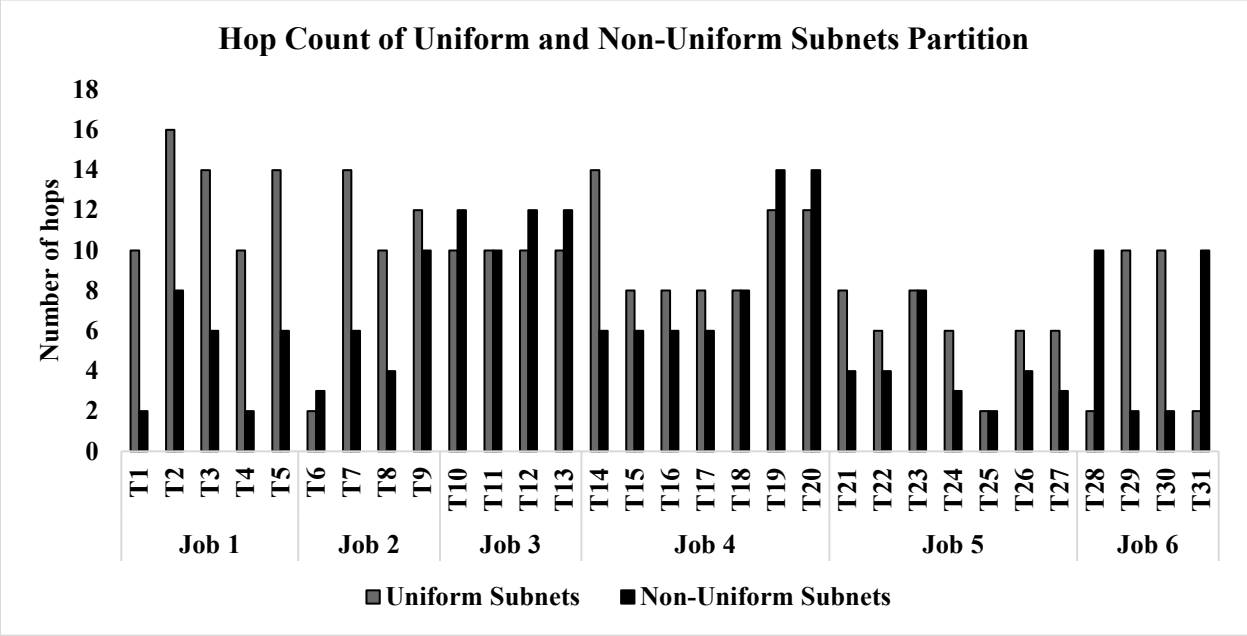


Figure 5.16: Hop count of uniform and non-uniform subnets in 64-core architecture

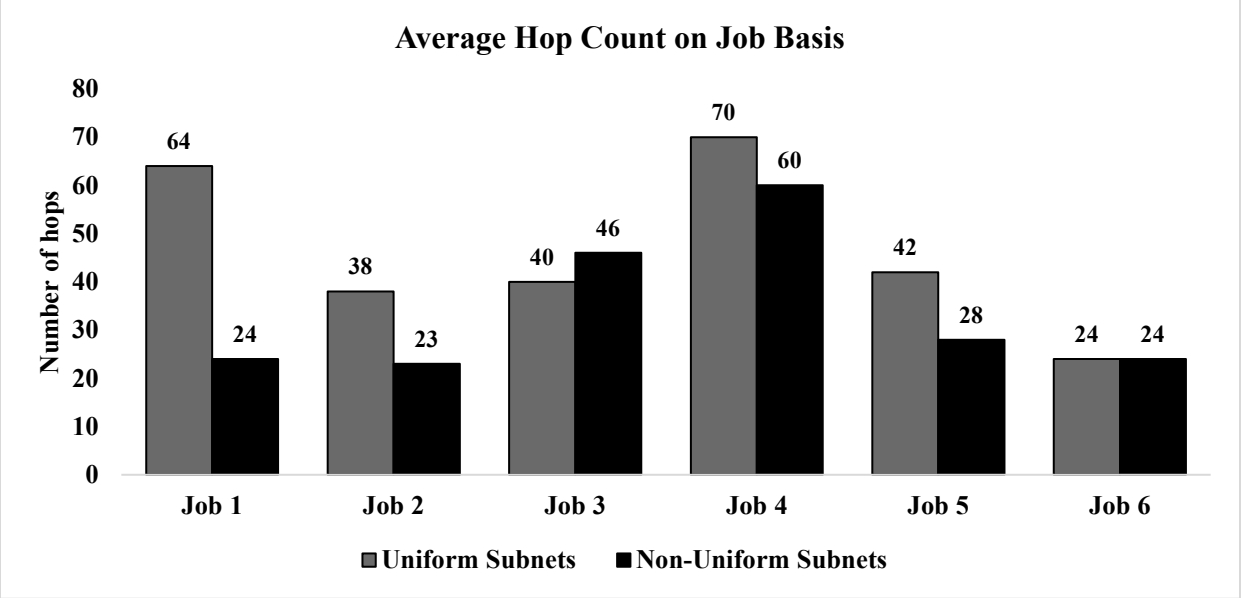


Figure 5.17: Average hop count on job basis

From Figure 5.18, it can be observed that the hop count due to the non-uniform 64-core architecture is reduced by 26.26% on average when compared with the uniform subnet

architecture. This is because the non-uniform subnets have close center core with distinct size of subnets in its architecture.

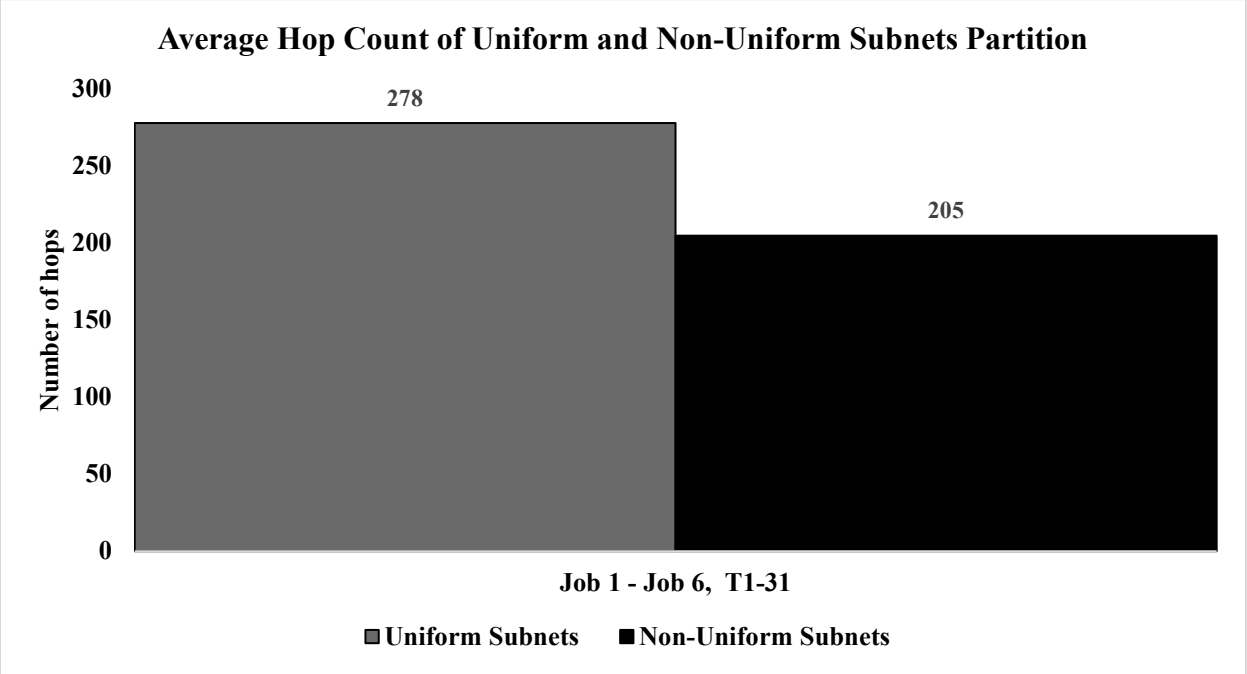


Figure 5.18: Average hop count of 64-core architecture

5.3.3 Power Consumption

Power consumption due to the uniform and non-uniform subnets 64-core architectures for all jobs is illustrated in Figure 5.19. The power consumption is different from one other even though both are WNoC-DDs architecture as they differ in logical partition. The change in subnet size varies the position of center core and neighbor cores. Thus, the routing path for jobs through individual tasks differ. The performance of the architecture will be considered best if the power consumption is reduced. The selection of any architecture mostly relies on latency and power consumption. The results of the architectures shown that non-uniform subnets perform well when compared to traditional uniform subnets.

Few in-subnet jobs may take more power as they traditionally follow mesh and updating the directories. However, as the subnet size varies, eventually power consumption in both architectures also varies.

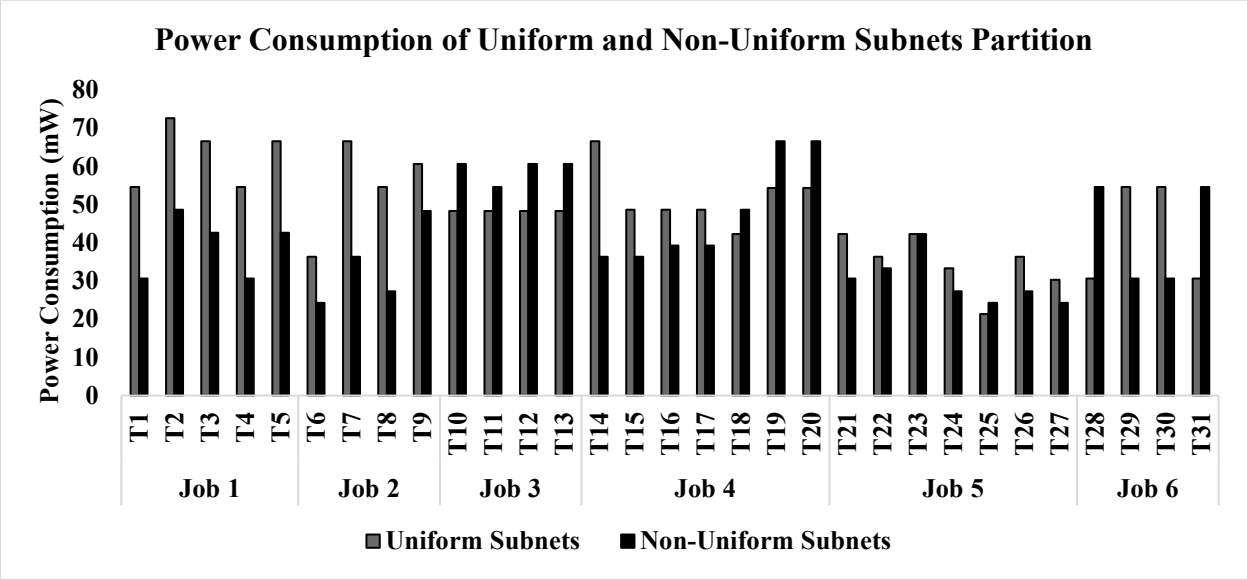


Figure 5.19: Power consumption of uniform and non-uniform subnets in 64-core architecture

Power consumption calculation is determined by the number of cores, wired-wireless routers, and directories involved in transferring the data between source and destination core. For every job, the above listed resources are considered and is illustrated in Figure 5.20. To make the calculations accurate and practically approved, the average of subnets is considered. Theoretically, the cores which are not active in any route are idle and they consume less power. However, by considering the average of the subnet is always a best case that is applicable in practical implementations. The performance calculation of power consumption according to job basis is similar to latency and hop count.

From Figure 5.21, it can be observed that the power consumption due to the non-uniform subnet architecture is reduced by 14.76% when compared with that of the uniform subnet architecture. This is because the center core directory and wireless router is assigned in unusual

positions. The dissimilar sizes of subnet prove that the performance is improved, and they are potential when considering other performance parameters too.

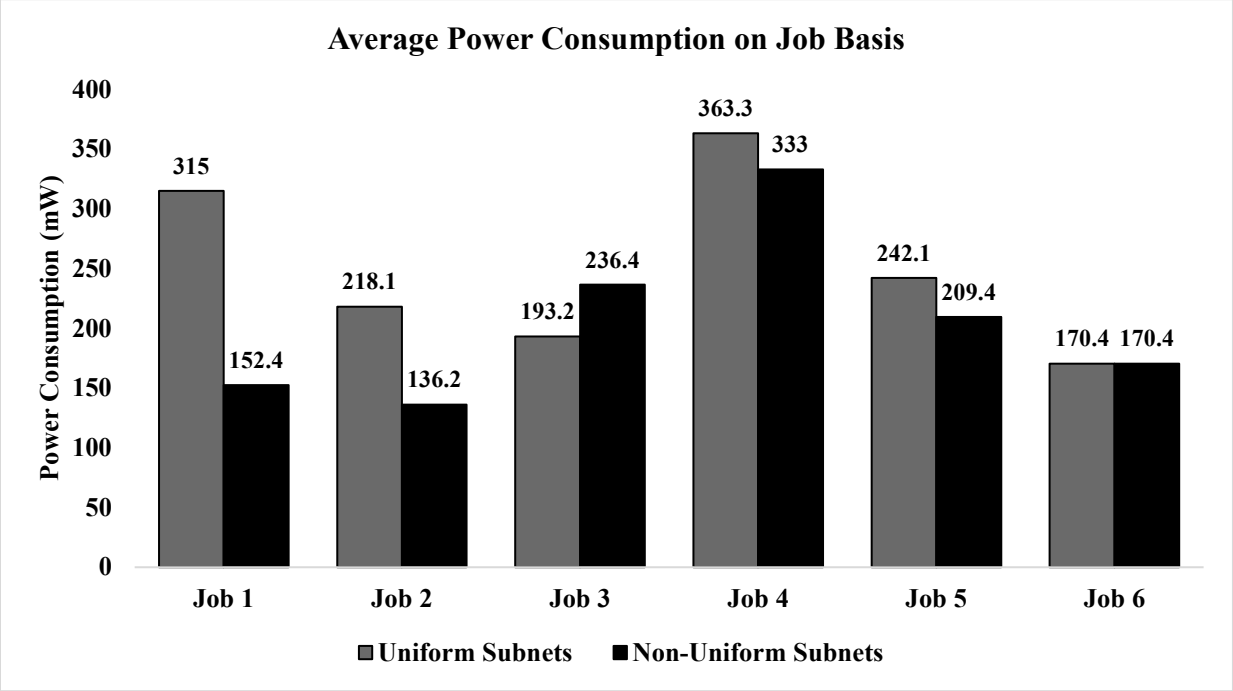


Figure 5.20: Average power consumption on job basis

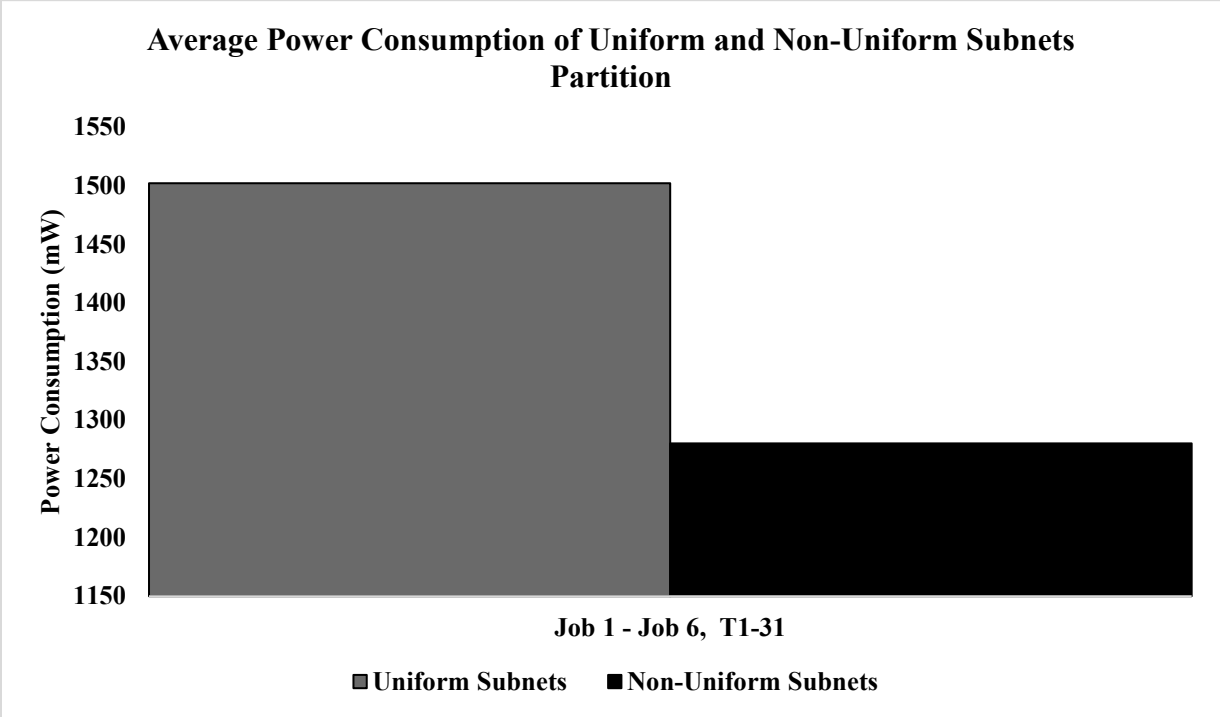


Figure 5.21: Average power consumption of 64-core architecture

CHAPTER 6

CONCLUSIONS AND FUTURE EXTENSIONS

In this research, we analyzed different architectures such as mesh, WNoC, WNoC-CD, WNoC-DD of 36-core capacity. Further, the research is extended to 64-core architecture. WNoC-DDs proved they are best when compared to other architectures. WNoC-CD architecture is not considered for 64-core as there are lot of bottlenecks such as traffic and subnet utilization to ensure performance. A single directory is not good enough to handle 64-core and so its workload.

6.1 Conclusions

Multicore architectures help improve performance to power ratio by concurrently using multiple cores. Contemporary multicore architectures have multilevel cache memory organization. Due to the presence of caches, multicore architectures suffer from high core-to-core communication latency and power consumption. Studies suggest that directory-based architecture with wireless routers has potential to decrease communication latency in multicore architectures. To address the cache coherence, latency and power consumption, we present a novel directory-based mechanism in WNoC architecture with a centralized directory (WNoC-CD) and wireless routers as proposed architecture 1. Instead of using the entire architecture for an application, uniform partition of subnets is introduced with a wireless router assigned to its center core, which helps in reducing the hops. We simulate a 2D mesh, traditional WNoC, and the proposed WNoC-CD architecture. According to the experimental results, the proposed architecture helps decrease the communication latency by up to 15.71% and the total power consumption by up to 67.58% when compared to the mesh architecture. Similarly, the proposed architecture helps decrease the communication latency by up to 10.01% and the total power consumption by up to 58.10% when

compared to the traditional WNoC architecture. The performance improvement in WNoC-CD architecture is by reducing the total number of hops.

However, they are several challenges with WNoC-CD architecture such as latency due to waiting time of tasks and for every out-subnet task the subnet must request the centralized directory. WNoC-CD architectures are not suitable for larger number of cores. To address the issues of WNoC-CD, we introduce a uniform subnet partition of WNoC architecture with distributed directories (WNoC-DDs) as proposed architecture 2. A directory allows the tasks to execute faster by providing adaptive minimal routing path to reach the destination node. VisualSim Architect is used to model and simulate the architectures by using synthetic workload and the workload is identical to proposed WNoC-CD. It is observed that the distributed directories significantly improve the performance of WNoC architecture, which supports the adaptability of WNoC-DDs to larger networks. With the proposed WNoC-DDs, individual subnets can operate simultaneously if/as the cores acquire the required data from its own subnet. As the individual directory maintains/tracks the status of other directories, it would take less time for processing without or any further queries for the required data. Experimental results show that the proposed WNoC-DDs reduces communication delay up to 20.54% and 5.40%, respectively, when compared to mesh and WNoC-CD. Similarly, the proposed WNoC-DDs reduces power consumption up to 73.56% and 19.97%, respectively, when compared to mesh and WNoC-CD. Finally, each of the distributed directories can control substantial number of cores compared to centralized directory.

With the increased number of cores, the performance may be improved but the complexities in coordinating with peer cores is always challenging. A 64-core architecture is considered with a different workload that has six jobs, which is divided into 31 subtasks. In WNoC architectures, the performance is mostly based on partition of subnets and the routing algorithms

followed. Uniform partition with increased number of cores leads to underutilization of cores, latency and power consumption due to the shift of center core. To address the issues of uniform partition, we introduce a non-uniform partition in WNoC-DDs as proposed architecture 3 to get advantage of getting closer center core in larger number of cores. The non-uniform partition is also satisfactory to assign subnets according to the workload such as number of cores required to complete the given job. The proposed technique can be applied to further large number of cores.

The designs and models are simulated using VisualSim tool. The tool allows to analyze the parameters and conveys useful information and provides trade-off performance of architectures. According to the experimental results, non-uniform WNoC-DD architectures helps in reducing the communication delay by up to 11.11%, hop count is reduced up to 26.26%, and the total power consumption by up to 14.76% when compared with the uniform subnets partition architecture. This is due to the selection of closer center cores and serving the subnets according to the range of cores required by a job. Thus, the introduction of non-uniform subnets is appropriate to address the issues of uniform subnets. Non-uniform subnets are with different subnet sizes and thus they give the opportunity of assigning workloads based on the subnet size. In such a way, the utilization is extended and reducing the latency, hop count, and power consumption.

6.2 Future Extensions

This work can be extended for future multicore/many-core system analysis. Some possible extensions are listed below:

- Designing, modeling, and simulating CPU-GPU architectures for big data analytics.
- Adding traffic parameters to the simulation of proposed architectures to check the range of workload that is enforced on a single hop and calculate bandwidth utilization and the range for each workload.

- Simulating multicore/many-core architectures to allow non-uniform subnets with dynamic working strategy to study performance.
- Study on 3D NoC architectures and investigate the impact of combining 3D routers with 3D processor architectures.

REFERENCES

[1] Meindl, James D, “Beyond Moore's law: The interconnect era,” *Computing in Science & Engineering*, Vol. 5, No. 1, pp. 20-24, 2003.

[2] Yeric, Greg, “Moore's Law at 50: Are we planning for retirement?,” In *Electron Devices Meeting (IEDM), 2015 IEEE International*, IEEE, 2015.

[3] Bambagini, Mario, Marko Bertogna, and Giorgio Buttazzo, “On the effectiveness of energy-aware real-time scheduling algorithms on single-core platforms,” In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pp. 1-8, IEEE, 2014.

[4] He, Liqiang, “Computer architecture education in multicore era: Is the time to change,” In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference* on, Vol. 9, pp. 724-728, IEEE, 2010.

[5] August, David, Keshav Pingali, Derek Chiou, Resit Sendag, and J. Yi Joshua, “Programming multicores: Do applications programmers need to write explicitly parallel programs?,” *IEEE Micro*, Vol. 30, No. 3, pp. 19-33, 2010.

[6] Atachiants, Roman, Gavan Doherty, and David Gregg, “Parallel performance problems on shared-memory multicore systems: taxonomy and observation,” *IEEE Transactions on Software Engineering*, Vol. 42, No. 8, pp. 764-785, 2016.

[7] Yu, Jie, Guangming Liu, Wenrui Dong, and Xiaoyong Li, “Using Locality-Enhanced Distributed Memory Cache to Accelerate Applications on High Performance Computers,” In Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and *IEEE International Conference on Intelligent Data and Security (IDS), 2017 IEEE 3rd International Conference* on, pp. 160-166, IEEE, 2017.

[8] Watanabe, Tadashi, “Future Technological Challenges for High Performance Computers,” In *PDCAT*, p. 2, 2005.

[9] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, et al., “Piranha: A scalable architecture based on single-chip multiprocessing,” in *Proceedings of the 27th International Symposium on Computer Architecture, Vancouver, B.C*, pp. 282-293, 2000.

[10] “From a few cores to many: A Terascale computing research overview,” Intel, 2006, http://download.intel.com/research/platform/terascale/terascale overview paper.pdf, (accessed on 3/15/2017)

[11] Xue, Yuankun, Zhiliang Qian, Guopeng Wei, Paul Bogdan, Chi-Ying Tsui, and Radu Marculescu. “An efficient network-on-chip (noc) based multicore platform for hierarchical parallel genetic algorithms,” In *Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium* on, pp. 17-24, IEEE, 2014.

[12] Psathakis, Antonis, Vassilis Papaefstathiou, Manolis Katevenis, and Dionisios Pnevmatikatos, "Design trade-offs in energy efficient NoC architectures," In *Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium* on, pp. 186-187, IEEE, 2014.

[13] Silva, Douglas RG, Bruno S. Oliveira, and Fernando G. Moraes, "Effects of the NoC Architecture in the Performance of NoC-based MPSoCs," In *Electronics, Circuits and Systems (ICECS), 2014 21st IEEE International Conference* on, pp. 431-434, IEEE, 2014.

[14] Rezaei, Amin, Farshad Safaei, Masoud Daneshtalab, and Hannu Tenhunen, "HiWA: A hierarchical wireless network-on-chip architecture," In *High Performance Computing & Simulation (HPCS), 2014 International Conference* on, pp. 499-505, IEEE, 2014.

[15] C. Wang, W. H. Hu, and N. Bagherzadeh, "A wireless network-on-chip design for multicore platforms," in *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 409-416, 2011.

[16] Morales, Luis Germán García, José Edinson Aedo Cobo, and Nader Bagherzadeh, "Simulation-Based Evaluation Strategy for Task Mapping Approaches in WNoC Platforms," In *Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference* on, pp. 622-626, IEEE, 2018.

[17] Shreedhar, Tanya, and Sujay Deb, "Hierarchical Cluster based NoC design using Wireless Interconnects for Coherence Support," In *VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference* on, pp. 63-68, IEEE, 2016.

[18] Joshi, Amit D., S. Indrajeet, N. Ramasubramanian, and B. Shameedha Begum, "Analysis of multi-core cache coherence protocols from energy and performance perspective," In *Recent Innovations in Signal processing and Embedded Systems (RISE), 2017 International Conference* on, pp. 381-388, IEEE, 2017.

[19] Shao, Y. Sophia, Sam Xi, Viji Srinivasan, Gu-Yeon Wei, and David Brooks, "Toward cache-friendly hardware accelerators," In *HPCA Sensors and Cloud Architectures Workshop (SCAW)*, pp. 1-6, 2015.

[20] Davis, John D., Zhangxi Tan, Fang Yu, and Lintao Zhang, "A practical reconfigurable hardware accelerator for Boolean satisfiability solvers," In *Design Automation Conference, DAC 2008, 45th ACM/IEEE*, pp. 780-785, IEEE, 2008.

[21] D. Lenoski, J. Laudon, K. Gharachorloo, W. D. Weber, et al., "The stanford dash multiprocessor," in *J. of Computer*, Vol. 25, No. 3, pp. 63-79, 1992.

[22] Cui, Jianqun, Yanxiang He, and Libing Wu, "More efficient mechanism of topology-aware overlay construction in application-layer multicast," In *Networking, Architecture, and Storage, 2007. NAS 2007. International Conference* on, pp. 31-36, IEEE, 2007.

[23] Schley, Gert, and Martin Radetzki, "Optimal distribution of privileged nodes in networks-on-chip," In *Intelligent Solutions in Embedded Systems (WISES), 2011 Proceedings of the Ninth Workshop* on, pp. 87-92, IEEE, 2011.

[24] Domke, Jens, Torsten Hoefler, and Satoshi Matsuoka, "Routing on the dependency graph: A new approach to deadlock-free high-performance routing," In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing,* pp. 3-14, ACM, 2016.

[25] Wettin, Paul, Ryan Kim, Jacob Murray, Xinmin Yu, Partha P. Pande, Amlan Ganguly, and Deukhyoun Heoamlan, "Design space exploration for wireless NoCs incorporating irregular network routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 33, No. 11, pp. 1732-1745, 2014.

[26] Shamim, Md Shahriar, Naseef Mansoor, Rounak Singh Narde, Vignesh Kothandapani, Amlan Ganguly, and Jayanti Venkataraman, "A wireless interconnection framework for seamless inter and intra-chip communication in multichip systems," *IEEE Transactions on Computers*, Vol. 66, No. 3, pp. 389-402, 2017.

[27] Yu, Zhiyi, "Towards High-Performance and Energy-Efficient Multi-core Processors," In *CMOS Processors and Memories*, pp. 29-51, Springer, Dordrecht, 2010.

[28] Pase, Douglas M., and Matthew A. Eckl, "A comparison of single-core and dual-core Opteron processor performance for HPC," *IBM xSeries Performance Development and Analysis*, 2005.

[29] H. V. Caprita, and M. Popa, "Design methods of multithreaded architectures for multicore microcontrollers," in *IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI),* pp. 427-432, 2011.

[30] J. M. Li, P. Jiao, and C. G. Men, "The Heterogeneous architecture of Multicore research and design," in *International Conference on Management and Service Science*, pp. 1-6, 2009.

[31] William Stallings, "Computer Organization and Architecture Designing for Performance," *8th edition, Prentice Hall, Pearson Publisher*, 2010.

[32] R. Jeyapaul, F. Hong, A. Rhisheekesan, A. Shrivastava, et al., "UnSync-CMP: Multicore CMP Architecture for Energy-Efficient Soft-Error Reliability," in *IEEE Transactions on Parallel and Distributed Systems,* Vol. 25, No. 1, pp. 254-263, 2014.

[33] Zhou, Qian, Yu-kun Song, Duo-li Zhang, and Gao-ming Du, "A design of multi-core system based on Avalon bus," In *Computer Science and Network Technology (ICCSNT), 2011 International Conference*, Vol. 3, pp. 1456-1459, IEEE, 2011.

[34] S. Bell, B. Edwards, J. Amann, R. Conlin, et al., "Tile64-processor: A 64-core soc with mesh interconnect," in *IEEE International Conference on Solid-State Circuits*, pp. 588-598, 2008.

[35] M. J. Saikia and R. Kanhirodan, "High performance single and multi-GPU acceleration for Diffuse Optical Tomography," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 1320-1323, 2014.

[36] Jadon, Shruti, and Rama Shankar Yadav, "Multicore processor: Internal structure, architecture, issues, challenges, scheduling strategies and performance," In *Industrial and Information Systems (ICIIS), 2016 11th International Conference* on, pp. 381-386, IEEE, 2016.

[37] Ahmed, Rana E., and Muhammad K. Dhodhi, "Directory-based cache coherence protocol for power-aware chip-multiprocessors," In *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference* on, pp. 001036-001039, IEEE, 2011.

[38] Lilja, David J, "Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons," *ACM Computing Surveys (CSUR),* Vol. 25, No. 3, pp. 303-338, 1993.

[39] Gilabert, F., Daniele Ludovici, Simone Medardoni, Davide Bertozzi, Luca Benini, and Georgi Nedeltchev Gaydadjiev, "Designing regular network-on-chip topologies under technology, architecture and software constraints," In *International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 681-687, IEEE, 2009.

[40] Ansari, Abdul Quaiyum, Mohammad Rashid Ansari, and Mohammad Ayoub Khan, "Performance evaluation of various parameters of network-on-chip (noc) for different topologies," In *India Conference (INDICON), 2015 Annual IEEE*, pp. 1-4, IEEE, 2015.

[41] Wang, Ling, Jianye Hao, and Feixuan Wang, "Bus-based and NoC infrastructure performance emulation and comparison," In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference* on, pp. 855-858, IEEE, 2009.

[42] DiTomaso, Dominic, Randy Morris, Avinash Karanth Kodi, Ashwini Sarathy, and Ahmed Louri, "Extending the energy efficiency and performance with channel buffers, crossbars, and topology analysis for network-on-chips," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, No. 11, pp. 2141-2154, 2013.

[43] Sadeghi, Mostafa, Amir Reshadi Nezhad, and Faezeh Memarzadeh Zavareh, "A new suggestion for improvement of mesh topology on NOC," In *Computer Science and Network Technology (ICCSNT), 2016 5th International Conference* on, pp. 667-671, IEEE, 2016.

[44] Dehyadgari, Masood, Mohsen Nickray, Ali Afzali-Kusha, and Zainalabein Navabi, "Evaluation of pseudo adaptive XY routing using an object oriented model for NOC," In *Microelectronics, ICM 2005, The 17th International Conference* on, IEEE, 2005.

[45] Roy, Abinash, Jingye Xu, and Masud H. Chowdhury, "Multi-core processors: A new way forward and challenges," In *Microelectronics, 2008. ICM 2008. International Conference* on, pp. 454-457, IEEE, 2008.

[46] Scott Mueller, and Mark Edward Soper, "Microprocessor Types and Specifications," June 8, 2001, http://www.informit.com/articles/article.aspx?p=130978&seqNum=4.

[47] Mori, Yosuke, and Kenji Kise, "The cache-core architecture to enhance the memory performance on multi-core processors," In *Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference* on, pp. 445-450, IEEE, 2009.

[48] Asaduzzaman, Abu, Mark P. Allen, and Tania Jareen, "An effective locking-free caching technique for power-aware multicore computing systems," In *Informatics, Electronics & Vision (ICIEV), 2014 International Conference* on, pp. 1-6, IEEE, 2014.

[49] Al-Waisi, Zainab, and Michael Opoku Agyeman, "An overview of on-chip cache coherence protocols," In *Intelligent Systems Conference (IntelliSys)*, pp. 304-309, IEEE, 2017.

[50] Daya, Bhavya K., Chia-Hsin Owen Chen, Suvinay Subramanian, Woo-Cheol Kwon, Sunghyun Park, Tushar Krishna, Jim Holt, Anantha P. Chandrakasan, and Li-Shiuan Peh, "SCORPIO: a 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium* on, pp. 25-36, IEEE, 2014.

[51] Cantin, Jason F., Mikko H. Lipasti, and James E. Smith, "Improving multiprocessor performance with coarse-grain coherence tracking," In *ACM SIGARCH Computer Architecture News*, Vol. 33, No. 2, pp. 246-257, IEEE, 2005.

[52] Patel, Avadh, and Kanad Ghose, "Energy-efficient mesi cache coherence with pro-active snoop filtering for multicore microprocessors," In *Proceedings of the 2008 international symposium on Low Power Electronics & Design*, pp. 247-252, ACM, 2008.

[53] Giri, Davide, Paolo Mantovani, and Luca P. Carloni, "NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators," In *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1-8, IEEE, 2018.

[54] Ahmed, Rana E., and Muhammad K. Dhodhi, "Directory-based cache coherence protocol for power-aware chip-multiprocessors," In *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference* on, pp. 001036-001039, IEEE, 2011.

[55] Asaduzzaman, Abu, and Kishore K. Chidella, "A novel directory based hybrid cache coherence protocol for shared memory multiprocessors," In *Phased Array Systems and Technology (PAST), 2016 IEEE International Symposium* on, pp. 1-6, IEEE, 2016.

[56] Nawinne, Isuru, Haris Javaid, Roshan Ragel, Swarnalatha Radhakrishnan, and Sri Parameswaran, "Exploring Multilevel Cache Hierarchies in Application Specific MPSoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 34, No. 12, pp. 1991-2003, 2015.

[57] Huang, Xiaoping, Xiaoya Fan, Shengbing Zhang, and Yuhui Chen, "DLWAP-buffer: A Novel HW/SW Architecture to Alleviate the Cache Coherence on Streaming-like Data in CMP," In *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium* on, pp. 23-28, IEEE, 2012.

[58] H. Xiao, T. Isshiki, H. Kunieda, Y. Nakase, et al., "Hybrid shared-memory and message-passing multiprocessor system-on-chip for UWB MAC," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 658-659, 2012.

[59] Shreedhar, Tanya, and Sujay Deb, "Hierarchical Cluster based NoC design using Wireless Interconnects for Coherence Support," In *VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference* on, pp. 63-68, IEEE, 2016.

[60] Cerutti, Isabella, Aman Mohammed Behredin, Nicola Andriolli, Odile Liboiron Ladouceur, and Piero Castoldi, "Ring versus bus topology: A network performance comparison of photonic integrated NoC," In *Transparent Optical Networks (ICTON), 2016 18th International Conference* on, pp. 1-4, IEEE, 2016.

[61] Pandey, Sujan, Manfred Glesner, and M. Muhlhauser, "On-chip communication topology synthesis for shared multi-bus based architecture," In *Field Programmable Logic and Applications, 2005 International Conference* on, pp. 374-379, IEEE, 2005.

[62] Jang, Yongho, Jungsoo Kim, and Chong-Min Kyung, "Topology synthesis for low power cascaded crossbar switches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 12, pp. 2041-2045, 2010.

[63] Cakir, Cagla, Ron Ho, Jon Lexau, and Ken Mai, "Modeling and design of high-radix on-chip crossbar switches," In *Proceedings of the 9th International Symposium on Networks-on-Chip*, p. 20, ACM, 2015.

[64] Mubeen, Saad, and Shashi Kumar, "Designing efficient source routing for mesh topology network on chip platforms," In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference* on, pp. 181-188, IEEE, 2010.

[65] Wang, Feng, Xiantuo Tang, Zuocheng Xing, and Hengzhu Liu, "UniMESH: The light-weight unidirectional channel Network-on-Chip in 2D mesh topology," In *Electronics, Communications and Computers (CONIELECOMP), 2015 International Conference* on, pp. 104-109, IEEE, 2015.

[66] Rohbani, Nezam, Zahra Shirmohammadi, Maryam Zare, and Seyed-Ghassem Miremadi, "LAXY: A Location-Based Aging-Resilient Xy-Yx Routing Algorithm for Network on Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 36, No. 10, pp. 1725-1738, 2017.

[67] Wang, Ling, Zhen Wang, and Yingtao Jiang, "A hybrid chip interconnection architecture with a global wireless network overlaid on top of a wired network-on-chip," In *System on Chip (SoC), 2012 International Symposium* on, pp. 1-4, IEEE, 2012.

[68] Kumar, T. Ananth, and R. S. Rajesh, "Towards power efficient wireless NoC router for SOC," In *Communication and Network Technologies (ICCNT), 2014 International Conference* on, pp. 254-259, IEEE, 2014.

[69] Liu, Chung-Hsin, and Chien-Yun Lo, "The study of WSN routing," In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pp. 422-428, ACM, 2009.

[70] Wang, Chifeng, Wen-Hsiang Hu, and Nader Bagherzadeh, "A wireless network-on-chip design for multicore platforms," In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference* on, pp. 409-416, IEEE, 2011.

[71] Zhang, Wang, Ligang Hou, Jinhui Wang, Shuqin Geng, and Wuchen Wu, "Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," In *Intelligent Systems, 2009. GCIS'09. WRI Global Congress* on, Vol. 3, pp. 329-333, IEEE, 2009.

[72] Kim, Ryan, Jacob Murray, Paul Wettin, Partha Pratim Pande, and Behrooz Shirazi, "An energy-efficient millimeter-wave wireless NoC with congestion-aware routing and DVFS," In *Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium* on, pp. 192-193, IEEE, 2014.

[73] Abadal, Sergi, Albert Mestres, Mario Nemirovsky, Heekwan Lee, Antonio González, Eduard Alarcón, and Albert Cabellos-Aparicio, "Scalability of broadcast performance in wireless network-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 12, pp. 3631-3645, 2016.

[74] Han, Xing, Yuzhuo Fu, Jiang Jiang, and Chang Wang, "A Subnetting Mechanism with Low Cost Deadlock-Free Design for Irregular Topologies in NoC-based Manycore Processors," In *Information Science and Control Engineering (ICISCE), 2016 3rd International Conference* on, pp. 110-114, IEEE, 2016.

[75] Wang, Xiaohang, Mei Yang, Yingtao Jiang, and Peng Liu, "On an efficient NoC multicasting scheme in support of multiple applications running on irregular sub-networks," *Microprocessors and Microsystems*, Vol. 35, No. 2, pp. 119-129, 2011.

[76] Reza, Akram, Midia Reshadi, Ahmad Khademzadeh, and Maryam Bahmani, "Norma: A hierarchical interconnection architecture for Network on Chip," In *Proceedings of the 3rd International Design and Test Workshop (IDT)*, pp. 5-10, 2008.

[77] Holsmark, Rickard, and Shashi Kumar, "An abstraction to support design of deadlock-free routing algorithms for large and hierarchical nocs," In *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference* on, pp. 59-66, IEEE, 2011.

[78] Singh, Jayant Kumar, Ayas Kanta Swain, Tetala Neel Kamal Reddy, and Kamala Kanta Mahapatra, "Performance evaluation of different routing algorithms in Network on Chip," In *Microelectronics and Electronics (PrimeAsia), 2013 IEEE Asia Pacific Conference on Postgraduate Research* in, pp. 180-185, IEEE, 2013.

[79] Ezz-Eldin, Rabab, Magdy Ali El-Moursy, and Hesham FA Hamed, "Network on Chip Aspects," In *Analysis and Design of Networks-on-Chip Under High Process Variation*, pp. 11-44, Springer, Cham, 2015.

[80] Morales, Luis Germán García, José Edinson Aedo Cobo, and Nader Bagherzadeh, "Simulation-Based Evaluation Strategy for Task Mapping Approaches in WNoC Platforms," In *Parallel, Distributed and Network-based Processing (PDP), 2018 26th Euromicro International Conference* on, pp. 622-626, IEEE, 2018.

[81] Asaduzzaman, Abu, Kishore K. Chidella, and Divya Vardha, "An energy-efficient directory based multicore architecture with wireless routers to minimize the communication latency," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 2, pp. 374-385, 2017.

[82] Chidella, Kishore K., and Abu Asaduzzaman, "A novel Wireless Network-on-Chip architecture with distributed directories for faster execution and minimal energy," *Computers & Electrical Engineering*, Vol. 65, pp. 18-31, 2018.

[83] Neishaburi, Mohammad Hossein, and Zeljko Zilic, "NISHA: A fault-tolerant NoC router enabling deadlock-free Interconnection of Subnets in Hierarchical Architectures," *Journal of Systems Architecture*, Vol. 59, No. 7, pp. 551-569, 2013.

[84] Chu, Slo-Li, Sheng-Jie Shu, Ching-Chung Chen, and Ching-Jung Chen, "Camellia: A Novel High Performance On-Chip Network for Multicore Architectures," In *Semantics, Knowledge and Grids (SKG), 2015 11th International Conference* on, pp. 186-191, IEEE, 2015.

[85] Saneei, Mohsen, Ali Afzali-Kusha, and Zainalabedin Navabi, "Low-latency multi-level mesh topology for NoCs," *The 18th International Confernece on Microelectronics (ICM)*, pp. 36-39, 2006.

[86] Ghosal, Prasun, and Tuhin Subhra Das, "L2star: A star type level-2 2d mesh architecture for noc," In *Microelectronics and Electronics (PrimeAsia), 2012 Asia Pacific Conference on Postgraduate Research* in, pp. 155-159, IEEE, 2012.

[87] O. Villa, D. P. Scarpazza, and F. Petrini, "Accelerating Real-Time String Searching with Multicore Processors," in *IEEE Computer*, Vol. 41, No. 4, pp. 42-50, 2008.

[88] Funabiki, Nobuo, Junki Shimizu, Toru Nakanishi, Kan Watanabe, and Shigeru Tomisato, "An Extension of Active Access-Point Selection Algorithm for Throughput Maximization in Wireless Mesh Networks," In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference* on, pp. 367-372, IEEE, 2011.

[89] Mamidi, Aditya Vamsi, Sarath Babu, and B. S. Manoj, "Dynamic Multi-hop Switch Handoffs in Software Defined Wireless Mesh Networks," In *Advanced Networks and Telecommuncations Systems (ANTS), 2015 IEEE International Conference* on, pp. 1-6, IEEE, 2015.

[90] Gharavi, Hamid, and Chong Xu, "Distributed application of the traffic scheduling technique for smart grid advanced metering applications using multi-gate mesh networks," In *Global Telecommunications Conference (GLOBECOM 2011)*, pp. 1-6, IEEE, 2011.

[91] Lankes, Andreas, Thomas Wild, and Andreas Herkersdorf, "Hierarchical NoCs for optimized access to shared memory and IO resources," In *Digital System Design, Architectures, Methods and Tools, 2009. DSD'09. 12th Euromicro Conference* on, pp. 255-262, IEEE, 2009.

[92] DiTomaso, Dominic, Avinash Kodi, Savas Kaya, and David Matolak, "iWISE: Inter-router wireless scalable express channels for network-on-chips (NoCs) architecture," In *High Performance Interconnects (HOTI), 2011 IEEE 19th Annual Symposium* on, pp. 11-18, IEEE, 2011.

[93] VisualSim Architect. Mirabilis Design. http://mirabilisdesign.com/new/visualsim/; 2016 [accessed on 6/20/17].

[94] Muhammad, Huda S., and Assim Sagahyroon, "Virtual prototyping and performance analysis of two memory architectures," *EURASIP Journal on Embedded Systems 2009*, No. 1, 2009.

[95] Asaduzzaman, Abu, Md Moniruzzaman, Kishore K. Chidella, and Perlekar Tamtam, "An efficient simulation method using VisualSim to assess autonomous power systems," In *SoutheastCon, 2016*, pp. 1-7, IEEE, 2016.

[96] Fang J, Lu J, She C, "Research on topology and policy for low power consumption of network-on-chip with multicore processors," In *Computational Science and Computational Intelligence (CSCI). IEEE. 2015 International Conference*, pp. 621-625, 2015.

[97] Biagetti G, Crippa P, Curzi A, Orcioni S, Turchetti C, "ToLHnet: A low-complexity protocol for mixed wired and wireless low-rate control networks," In *Education and Research Conference (EDERC). IEEE. 2014 6th European Embedded Design*, pp. 177-181, 2014.

[98] Mondal HK, Deb S, "An energy efficient wireless Network-on-Chip using power-gated transceivers," In *System-on-Chip Conference (SOCC), 2014 27th IEEE International*, pp. 243-248, 2014.

[99] Deb S, Ganguly A, Pande PP, Belzer B, Heo D, "Wireless NoC as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 2, No. 2, pp. 228-39, 2012.

[100] Johari S, Sehgal VK, "Master-based routing algorithm and communication-based cluster topology for 2D NoC," *The Journal of Supercomputing*, Vol. 71, No. 11, pp. 4260-86, 2015.

[101] Chawade SD, Gaikwad MA, Patrikar RM, "Review of XY routing algorithm for network-on-chip architecture," *International Journal of Computer Applications*, Vol. 43, No. 21, 2012.

[102] Wang, Xiaofang, "A novel on-chip interconnection topology for mesh-connected processor arrays," In *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium* on, pp. 450-451, IEEE, 2010.