An Energy-Efficient Directory Based Multicore Architecture with Wireless Routers to Minimize the Communication Latency

Abu Asaduzzaman, Kishore K. Chidella, and Divya Vardha

Abstract—Multicore architectures suffer from high core-to-core communication latency primarily due to the cache's dynamic behavior. Studies suggest that a directory-approach can be helpful to reduce communication latency by storing the cached block information. Recent studies also indicate that a wireless router has potential to help decrease communication latency in multicore architectures. In this work, we propose a directory based multicore architecture with wireless routers to minimize communication latency. We simulate systems with mesh (used in the Standford Directory Architecture for SHared memory (DASH) architecture), wireless network-on-chip (WNoC), and the proposed directory based architecture with wireless routers. According to the experimental results, our proposed architecture outperforms the WNoC and the mesh architectures. It is observed that the proposed architecture helps decrease the communication delay by up to 15.71 percent and the total power consumption by up to 67.58 percent when compared with the mesh architecture. Similarly, the proposed architecture helps decrease the communication delay by up to 10.00 percent and the total power consumption by up to 58.10 percent when compared with the WNoC architecture. This is due to the fact that the proposed directory based mechanism helps reduce the number of core-to-core communication and the wireless routers help reduce the total number of hops.

Index Terms—Communication latency, directory based multicore architecture, energy-efficient multicore architecture, mesh topology, wireless network-on-chip architecture, wireless router

INTRODUCTION

In the present era, the computer has become a pervading electronic component from the beginning to the end of the day, by providing large number of platforms for computing, research, and entertainment. The contemporary usage of computer applications presents many challenges for design engineers. In order to provide improved performance to customers, design engineers have to change from building a single large complex core to many small simple cores on the computer chip [1], [2], [3], [4], [5]. Multicore chips have become the most predominant components in building a computer, which can meet the demands of the present time and the evolution of these multicore processors has made a radical change in the way of designing computer systems.

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 [6]. 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. In a multicore system, multitasking is done by assigning different sets of tasks to different cores. Multiple cores help improve the computational capacity of the processor through parallel computing technology [7]. While multicore processors have become certainly important, there are still many obstacles that designers have to face while designing more than one processing core on a chip. More cores were embedded 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 (some of those are discussed in this article).

A wide changeover from single-core to multicore chip processors has pleased many users by providing increased performance and access to many applications which requires a very fast execution speed. Multiple cores on a chip lead to some interesting challenges, such as implementing multi-threading

in order to provide better performance. Multi-threading is a process in which a central processing unit (CPU) can execute several number of threads simultaneously [8]. Simultaneous multithreading (SMT) is a technique in which several independent threads would issue instructions to a superscalar's multiple functional units in a single cycle [9]. Therefore, the advanced multicore chip multiprocessor (CMP) supports a number of cores (10 to 100 or more) on a chip. Examples on CMP include Tilera's 64-core TILE64 [10], Intel's 80-core TFLOPS [11], NVIDIA's 128-core Quadro GPU [12], 240-core Tesla C1060 GPU [12], and 2880-core Tesla K40 GPU [13].

With the evolution of multicore architectures, both simultaneous execution of instructions and better communication among the cores are given equal priority to provide outstanding performance. Coherence of cached data may be a major challenge as one large common memory is shared among many cores in a multicore architecture. There are various solutions to address multicore data inconsistency [14], [15], [16], [17].

Many multicore chip processor computers are capable of providing an adequate performance by implementing additional instruction level parallelism (ILP). ILP is implemented to keep cores productive as much as possible by executing several instructions at a time. This process requires access to memory in order to fetch data and store results. Therefore, many operations may be carried out on a particular memory location at a single instance of time. Data coherence mechanism ensures that the data fetched by a particular core is the most appropriate data. Each core on a multicore processor should be provided access to sufficient resources at any particular instant of time for a successful outcome. Also, some cores on the chip may depend on other cores because a number of execution units may require results computed by the other cores; therefore, inefficient communication among cores could result in a considerable delay in overall execution of some instructions [18]. With the implementation of additional parallelism and multithreading, efficient communication among cores on chip becomes central focus to achieve better performance [19]. In order to keep all the computing units (i.e., cores) active, these cores must be supplied with data at the instance of time when the data is requested, without any delay. Communication among cores can be viewed in two different perspectives: one is with respect to physical infrastructure (i.e., how cores communicate) and the other one is what is being transferred between the cores [20].

Designing of physical infrastructure deals with the resources and wiring, necessary to facilitate efficient transfer of the bits. There are many communication patterns that are proposed for efficient on-chip communication, but there are certain constraints to be considered, such as limited area, communication latency, and power consumption, while proposing a mechanism for communication between cores on-chip [21]. As more and more cores are tightly coupled in a multicore architecture, there is a necessity for the coupling of an efficient on-chip network with a proper design of coherence protocol which results in a synergistic relationship between the two. If two cores are placed on a single die instead of one, it should consume more power than that by a single core and generates large amount of heat. 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. In some proposed architectures, multiple cores are run at a very lower frequency to reduce power consumption.

As we know, bus based multicore architecture is okay for small number of cores (say, 4-8). Usage of dedicated wires to directly connect cores on-chip is one solution to provide direct communication in a multicore architecture. However, the manufacturing of chips using dedicated wires is very expensive and would consume more power and space; this inefficiency resulted in a shift to on-chip networks and incorporating wireless communication among cores. Network-on-chip (NoC) provides a more scalable solution for the multicore architectures [22]. NoC architecture is a technology, proposed to overcome the problem of large communication delay among cores in a multicore architecture. It is proposed in order to design a communication subsystem between various modules such as core processors, memories, and

special core blocks. Communication among these modules would be via multiple point-to-point links so that each and every block can reach any other module by switching over several links.

Many alternative communication patterns have been proposed, such as optical interconnections, radio frequency interconnect (RF-I) transmission lines, and complementary metal-oxide semiconductor (CMOS) ultra-wideband (UWB) wireless interconnect technology [23], [24], [25], [26]. The main objective of these architectures is to reduce transmission latency and power dissipation. However, challenges such as designing of an efficient transmitter and receiver and high manufacturing costs prevent these architectures from becoming a feasible solution. A new platform for multicore architecture is proposed which overcomes these limitations by using the CMOS UWB wireless interconnections which enables on-chip multi-hop communications by embedding wireless communication channels between cores in a multicore architecture [1], [27], [28], [29]. The effect of long distance transmission is reduced by using express channels [27], [28].

In the present design of a multicore architecture, many routing algorithms have been proposed to address the problem of communication delay. A routing algorithm determines the path through which data has to travel, it can be deterministic or adaptive, and it can be further classified as minimal or non-minimal routing algorithms. A deterministic algorithm is one which is predetermined, allows only one fixed path from the requesting core to the destination core. Whereas the adaptive routing selects one path from the multiple numbers of paths between source core and destination core and focuses on the current state of the network. Minimal routing algorithms, as the name implies, chooses the shortest path between the source and destination cores while the non-minimal routing algorithm may include some additional number of hops.

Some multicore architectures use dimension routing (i.e., X-Y routing algorithm) in order to reach the destination core from the requesting core. WNoC architecture uses an adaptive minimal routing algorithm which takes buffer utilization into account when deciding the routing path. Buffer utilization has been proven to achieve better performance than conventional routing [29]. The topology of a multicore architecture on chip plays a predominant role in the communication latency. In this paper, we propose a directory based architecture with wireless routers in order to reduce communication latency.

The rest of the paper is organized as follows: Section 2 summarizes some related published articles. The proposed directory based multicore architecture with wireless routers is introduced in Section 3. The experimental details are described in Section 4. Some experimental results and related discussions are presented in Section 5. Finally, this work is concluded in Section 6.

LITERATURE SURVEY

This section provides an overview of interconnection topologies including mesh and WNoC which are suitable for multicore architectures. Standford DASH multiprocessor architecture is also studied as we propose a directory based mechanism in this work.

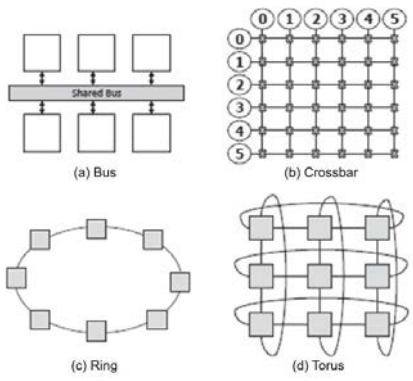

Fig. 1. Different popular interconnection network topologies.

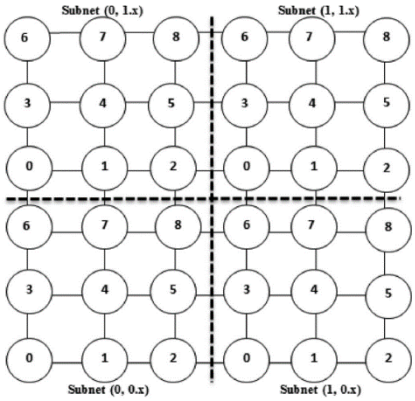

Fig. 2. Subnets in a mesh.

Interconnection Network Topologies

Some popular network topologies (namely, bus, crossbar, ring, and torus) are depicted in Fig. 1 [20], [30], [31].

The emphasis on designing new on-chip interconnection network topology is increasing as more and more cores have to be accommodated on a single chip. Using dedicated ad-hoc wires to connect cores and related components is an alternative for building an on-chip network. However, as the number of cores increases, the number of wires required to connect all the cores and components will become expensive. Moreover, scaling becomes a very significant problem along with massive power consumption. The length of the dedicated wires used to connect results in long latency.

NxN Mesh Network

Mesh (a simplified form of torus) is a potential network topology for multicore architectures. In a two dimensional (2D) mesh network, all cores are connected in a crossbar connection as shown in Fig. 2. Mesh network topology is the most common topology used, due to its advantages of shorter wavelength, low router complexity, and feasibility [11]. Wired mesh network provides very good reliability for intercore communication. However, this topology has many disadvantages, such as network congestion, poor scalability, high power consumption, and long latency. There are many routing algorithms used by various topologies in order to reach the destination core.

Wireless Network-On-Chip

In a multicore CMP, NoC is considered as the most prominent technology providing a high degree of integration among the cores. The 2D floor plan limits the choice of NoC topology. Express channels are introduced to bridge the gap among the cores which are far apart [27], [28]. Proposed 3D NoCs has advantages from both 3D integrated circuits (ICs) and NoCs and it provides improvements in latency and power consumption [32].

An UWB NoC technology is proposed utilizing both wired signals and distributed medium access control (MAC) protocol [24]. The proposed architecture achieves a data transmission range of 1 mm with the antenna length of 2.98 mm and bandwidth achieves on a single channel was 10 Gbps. As the range of wireless radio frequency signals are in the range of GHz, and the sizes of transmitter, receiver, and antenna blocks have also reduced significantly. A scalable wireless interconnect technology is proposed to evaluate the feasibility of both hybrid wired and wireless platform by validating on-chip wireless communication [22]. This work provides a two-tier hybrid wired and wireless architecture platform to demonstrate the benefits of long-range wireless links.

WNoC Topology

The basic component in a WNoC architecture is a network based processor array (NePA). Each and every processing element (PE) consists of a processor core, network interface (NI), and a router. The architecture of a NePA is shown in Fig. 3. Each of these routers 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. Fig. 4 shows the input and output ports of a NePA router.

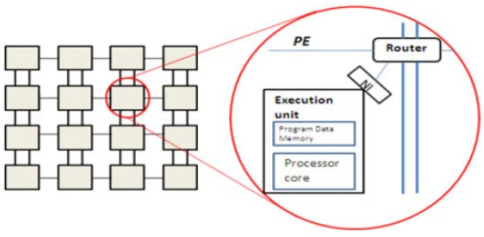

Fig. 3. A 4 x 4 2D NePA architecture.

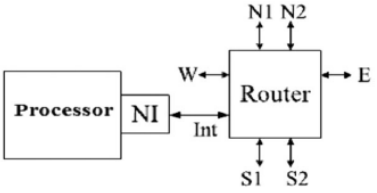

Fig. 4. NePA router port description.

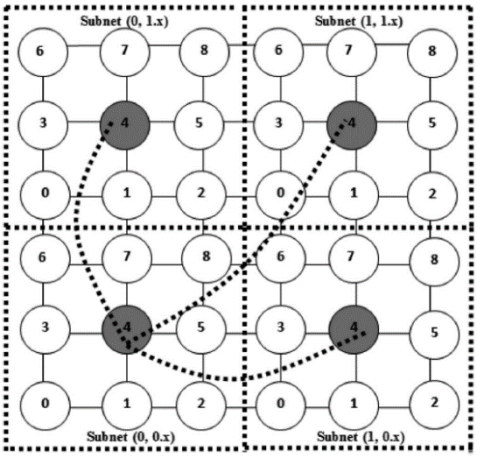

Fig. 5. Wireless network on-chip (WNoC) architecture.

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 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 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. Therefore, WNoC is capable of transferring packets through wired and wireless links. In WNoC, processing cores are divided into various subnets which have one wireless router responsible for providing wireless communication for the cores. Fig. 5 shows the architecture of a WNoC, where the cores are divided into four rectangular subnets and the wireless routers are placed in the middle (approximately) of each subnet. 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 and every node is identified within its subnet using a local address. The address has three components (as shown in Fig. 5) – 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 features of addressing a specific core in a network help WNoC provide much faster routing decisions as well as a scalable hierarchical system.

DASH (Directory Architecture for Shared Memory) Multiprocessor

Stanford DASH architecture is considered in this work 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 [33].

This architecture provides high processor performance by maintaining coherence among the caches of all the cores and also provides scalability of cores since 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 [34].

Fig. 6 shows the high level organization of a DASH system [35]. Typically, a DASH system may consist of a large number of processing nodes via an interconnection network which has large bandwidth and a low communication delay. The physical memory or the main memory is distributed among all the clusters in such a manner that the memory is accessible from each and every core.

Each processing core has its own individual cache. In order 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.

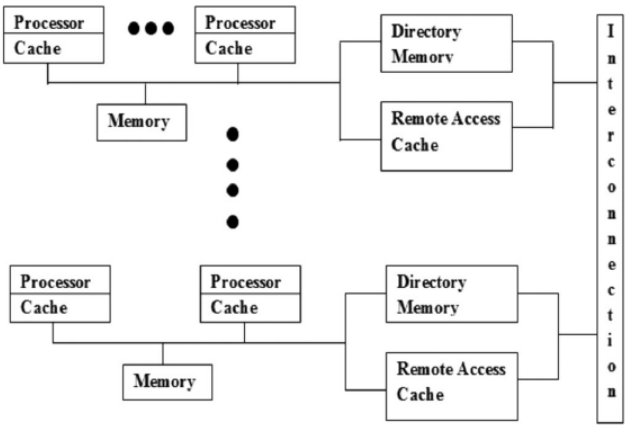

Fig. 6. A directory based DASH system.

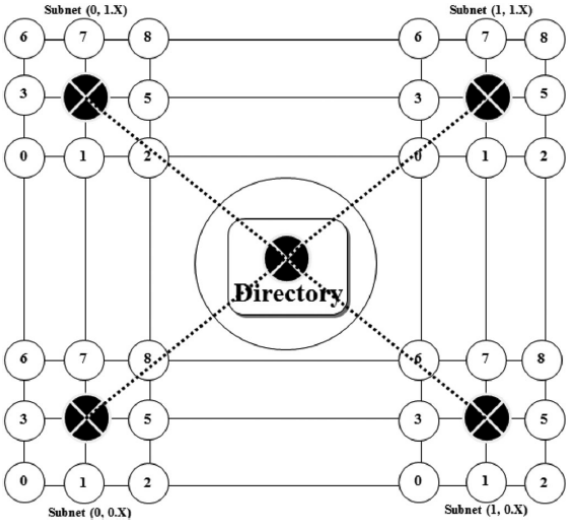

Fig. 7. Proposed architecture with directory and wireless routers.

PROPOSED DIRECTORY BASED ARCHITECTURE

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 core to a destination core using the directory and wireless routers. The key design considerations include: grouping cores, designing directory, managing cache consistency, and communication among cores.

Grouping Cores

Considering the WNoC and DASH architectures, the proposed architecture divides the cores on the die into clusters, called subnets, as shown in Fig. 7.

In each subnet, one special core (such as Core 4 in Subnet (0, 0.x) in Fig. 7) contains both wireless and wired routers and the other cores contain wired routers. Considering a 6 x 6 mesh topology, the cores are grouped into 3 x 3-core subnets, forming four quadrants. A new special core is introduced in the center to hold the proposed directory. All the cores inside a subnet are local to the subnet and the cores outside of a sub-net 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 a number of memory blocks in its cache to accommodate the data fetched from the main memory.

Designing Directory

A powerful processor (similar to the Power Processing Element in a Cell processor [36]) with a wireless router is used to host the directory. The directory contains cores' subnet addresses, the status of each cached block (if any), 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 (1,024 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 = 1,024 Bytes /128 Bytes = 8. Therefore, for an n-core system, n x (1 + 8) entries are required. (There should be 'n' rows for n columns. In each row, one column for the core number and eight columns for eight blocks.) Table 1 shows one row for an 8-core system. Initially, the blocks for each core in the directory will be empty. Whenever a core caches data, the particular block address of the specific data is recorded to the corresponding block of that core. This technique is explained later using Tables 1, 2, and 3.

It should be noted that proper cooling mechanism is required to avoid any hotspot due to a large number of activities in the centralized directory.

TABLE 1
A Row In Directory Representing Initial Stage For 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 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
A Row in Directory Showing Changes for Write in a Block

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	M 100th Blk	0 Addr Empty	0 Addr Empty	0 Addr Empty

Directory Mechanism

The directory mechanism 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 1 KB cache with 128 B lines:

1. Initially all the blocks of the directory would be empty with Status '0' (as shown in Table 1)
2. After Core-1 makes a request to fetch 100th block of the next level cache/memory, the block number is 100% 8 ¼ 4. The fetched data is stored in the fourth block of the cache of Core-1 (see Table 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 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 shown in Table 3). A protocol to manage cache consistency is explained next.

Managing Cache Consistency

As we know, the cache coherence protocol deals with the consistency among all caches by maintaining memory coherence. The well established MESI (short for Modified, Exclusive, Shared, and Invalidate) protocol from the University of Illinois at Urbana-Champaign [37] is used for managing cache consistency of the proposed architecture. Initially, the cache (of all cores) is empty – the states in the directory are indicated with '0' and the address fields are empty. With respect to the read/write requests, the state of a block in the directory changes accordingly as illustrated in Tables 1, 2 and 3. The directory keeps track of each cached block and maintains its state. For example, when a core requests for a read operation for the first time, the data of that particular 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 particular block is changed to 'M' (for Modified). For every write operation, the directory is updated with an 'I' (for Invalidate) for other cached copies. The state 'S' (for Shared) of a cached block means more than one cores are sharing that particular block.

Communication Among Cores

Cores inside a subnet communicate via the wired network. To communicate with cores in different subnets, directory and wireless routers are used. A directory is implemented in a special powerful core as shown in Fig. 7 (the core in the center). The directory quickly provides information regarding the status and address of a block cached by cores (if any). 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.

The search process is confined to the directory core. For each and every request made by a core, a unique thread should be generated, which is independent of the other threads generated by other cores. The central directory core should be capable of performing simultaneous multithreading [38] in order to serve all the requests simultaneously, as soon as they arrive at the directory core, without having the requests wait in a queue for long time.

Many communication cases are possible in a 6 x 6 WNoC. Table 4 shows 25 different communication cases that are considered in this study. We consider some cases where communication is done between two cores in two different subnets, such as Case 1 (communication between Core (0, 0.0) and Core (1, 1.8)). We also consider cases where communication is done between two cores in the same sub-net, such as Case 25 (communication between Core (0, 1.7) and Core (0, 1.1)). The number of hop counts from the source core to the destination core is the count of number of cores that the request has to traverse to reach the destination core. We consider the communication delay as the delay caused by all the intermediate and destination cores to process the header information and route the requested packet. Power consumption in the process of inter-core communication is evaluated for all cases in different architectures. In order to evaluate the performance of the proposed architecture, we consider all three systems mentioned earlier, namely mesh, WNoC, and the proposed directory based architecture with wireless routers.

EXPERIMENTAL DETAILS

We evaluate our work (a wireless NoC system with the proposed directory) by comparing it with the popular mesh topology and WNoC architecture. We use VisualSim (a world-class modeling and simulation platform suitable for almost all computing systems) [39] tool to model and simulate the systems. In this section, we describe the experimental details including assumptions and workload. Then we explain how latency, hop count, and power consumption are calculated.

Assumptions

It is assumed that the properties of the cores (except the central directory-core of the proposed system) and wires in all three architectures (i.e., mesh, WNoC, and "proposed") are same. Also assumed that the wireless routers used in the WNoC and proposed architectures are identical. Multicore systems with a small number of nodes (about 36) are considered so that the performance of the architectures can be observed closely. Same workload is used to run the simulation programs of different systems. 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 [19]. 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). There-fore, if the delay due to an intermediate core is 4 units, the delay caused due to a destination core is assumed to be 40 units.

TABLE 4
Source and Destination Nodes for Different Communication Cases

Case Number	Source Node (S)	Destination Node (D)	Remark
Case 1	Core - > (0, 0.0)	Core - > (1, 1.8)	S and D are in different subnets
Case 2	Core - > (0, 0.4)	Core - > (1, 1.4)	S and D are in different subnets
Case 3	Core - > (0, 0.7)	Core - > (1, 0.1)	S and D are in different subnets
Case 4	Core - > (0, 0.3)	Core - > (0, 1.5)	S and D are in different subnets
Case 5	Core - > (1, 0.5)	Core - > (0, 1.2)	S and D are in different subnets
Case 6	Core - > (1, 0.7)	Core - > (0, 1.5)	S and D are in different subnets
Case 7	Core - > (0, 1.0)	Core - > (1, 0.0)	S and D are in different subnets
Case 8	Core - > (0, 0.8)	Core - > (1, 1.6)	S and D are in different subnets
Case 9	Core - > (0, 0.7)	Core - > (0, 1.1)	S and D are in different subnets
Case 10	Core - > (1, 1.5)	Core - > (1, 0.2)	S and D are in different subnets
Case 11	Core - > (0, 1.3)	Core - > (0, 0.1)	S and D are in different subnets
Case 12	Core - > (0, 1.4)	Core - > (1, 0.6)	S and D are in different subnets
Case 13	Core - > (1, 0.1)	Core - > (1, 1.1)	S and D are in different subnets
Case 14	Core - > (1, 1.2)	Core - > (0, 0.8)	S and D are in different subnets
Case 15	Core - > (1, 0.6)	Core - > (0, 1.2)	S and D are in different subnets
Case 16	Core - > (1, 0.4)	Core - > (0, 1.7)	S and D are in different subnets
Case 17	Core - > (1, 1.3)	Core - > (0, 1.3)	S and D are in different subnets
Case 18	Core - > (0, 1.2)	Core - > (1, 1.0)	S and D are in different subnets
Case 19	Core - > (0, 0.1)	Core - > (1, 0.7)	S and D are in different subnets
Case 20	Core - > (1, 0.2)	Core - > (0, 1.6)	S and D are in different subnets
Case 21	Core - > (0, 0.6)	Core - > (0, 0.5)	S and D are in the same subnet
Case 22	Core - > (1, 0.7)	Core - > (1, 0.8)	S and D are in the same subnet
Case 23	Core - > (0, 1.4)	Core - > (0, 1.2)	S and D are in the same subnet
Case 24	Core - > (1, 1.6)	Core - > (1, 1.2)	S and D are in the same subnet
Case 25	Core - > (0, 1.7)	Core - > (0, 1.1)	S and D are in the same subnet

Workload Used

Workload is used in the experiments to run the simulation programs and calculate latency, hop count, and power consumption. We consider 25 different cases as shown in Table 4. The cases 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 tool.

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. Cases 1 through 20 can be better executed by WNoC architecture. In 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. The scenarios such as Case 9 and Case 18 prove the deficiency of such systems.

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 the proposed directory system. The proposed system 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 nodes are considered based on the activities of the subnets. Thus, the directory should help balance load by selecting the destination nodes from different subnets (if possible).

As shown in Table 5, for some cases (such as Cases 8 and 9) WNoC takes more time than mesh, for some cases (such as Case 10) WNoC and mesh take same amount of time, and for some cases (such as Cases 1, 2, and 20) WNoC takes less time than mesh. However, for most cases the proposed system takes less time than the WNoC (and mesh).

TABLE 5
Communication Latency for Three Different Architectures

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

Hop Count

In case of mesh multicasting, distance between two cores is one hop count if the two cores are directly connected to each other. In order to transfer the information in mesh multicasting (and WNoC architecture), the status of the block is required to fetch the data from destination core. To ensure that the communication is successful in mesh architecture, return path or acknowledgement is essential. So, the total number of hops (HT) is 2x the number of hops between the source and destination. The hop count of WNoC has similar drawback because of its routing methodology for outer subnets.

However, the proposed architecture does not require any acknowledgement path for identifying the status as well as fetching the data. The directory works as a commander and supervises the purpose without any acknowledgement. The routing path has become straight forward and less due to the introduction of directory in a multicore architecture. So, the total hops from source to directory are simplex and requires only about one half of the hops compared to mesh and WNoC architectures in an average. Table 6 shows the hop counts for the mesh, WNoC, and proposed architectures. For all cases (except Case 15), the hop count for the proposed system is smaller than that for the WNoC (and mesh).

Power Consumption

The considerations and assumptions that are made for exploring the power consumption of the three simulated architectures are listed in Table 7. It is assumed that each wired link consumes 1 unit of power (P_{wr}). Studies indicate "a wired network connection would take less power than a wireless network" [40], [42]. 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 3 units of power (P_{cwr}). The XY routing algorithm does not have a unique pattern path towards destination, so the average power consumed by a core in a 6 x 6mesh (P_{canw}) is average number of cores travelled x power needed for each wired core (19.5 units). Similarly, the average power consumed by a wired link in a 6 x 6 mesh (P_{alwr}) is 5.5 power units.

For the WNoC architecture, each wired core consumes 3 units of power (similar to a core in mesh) and the special core with wireless router consumes 3.3 units of power (P_{cwl}). For a 3 x 3 subnet, the minimum number of links is one and the maximum number of links is four. So the aver-age power consumed by a wired link in a subnet (P_{awrsn}) is 2.5 power units. The average power consumed by a core in a subnet (P_{casn}) is 7.5 units of power.

In the proposed architecture, the power consumed by the directory (P_{dr} = 6 units) is assumed to be twice the power consumed by the core, since the whole directory has to be scanned in a worst scenario. Similarly, the power consumed by the directory-core with wireless router (P_{cdr} = P_{dr} + power needed by a wireless core = 6 + 3.3) is 9.3.

The total power consumed by a system depends on the hop count and the routing methodology. The reduced hop count due to the proposed architecture should minimize the power consumption and offer better performance when compared with mesh and WNoC architectures. The power consumption for a particular inside-subnet case is similar for WNoC and the proposed architectures (see Cases 21-25 in Table 8).

RESULTS AND DISCUSSION

In this section, we present some experimental results. Fig. 8 illustrates the communication latency due to the mesh, WNoC, and proposed architectures for all 25 cases. For most cases (except Cases 9, 13, 15, and 18), the latency due to the proposed architecture is smaller or same compared to that due to the mesh and WNoC architectures.

For Cases 9, 13, 15, and 18, the cores are next to each other, so mesh provides the least delay. In the WNoC and proposed architectures, the transmitting path will be a broadcasting technique for the destination out of its own subnet. Considering all cases, it is observed that the proposed architecture helps

reduce the communication latency, in an average, by 15.71 percent compared to mesh architecture and 10.00 percent compared to WNoC architecture.

The hop counts due to the mesh, WNoC, and proposed architectures for all 25 cases are shown in Fig. 9. It is observed that the hop count due to the proposed architecture is reduced by 67.45 percent when compared with the mesh architecture and 59.41 percent when compared with the WNoC architecture. This is because the proposed architecture does not require any acknowledgement or any return path to the source to complete the task. To process any request irrespective of destination subnet, the directory handles the request and ensures delivery.

TABLE 6
Hop Counts for the Mesh, WNoC, and Proposed Architectures

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

TABLE 7
Consideration and Assumptions for Power Calculation

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 whole path in WNoC between source and destination	P_{sd}	(vary)
10	Power consumed by whole path in WNoC between destination and source	P_{ds}	(vary)
11	Power consumed by a core with a wireless router	P_{cwl}	3.3
12	Power consumed by the wired links in a subnet on an average- WNoC	P_{awrsn}	2.5
13	Power consumed by the cores in a subnet on an average- WNoC	P_{casn}	7.5
14	Power consumed between source and directory-"proposed"	P_{sdr}	(vary)
15	Power consumed by the directory-"proposed"	P_{dr}	6.0
16	Power consumed by the directory core in "-proposed"	P_{cdr}	9.3

Fig. 10 illustrates the power consumption due to the mesh, WNoC, and proposed architectures for all the cases. According to the experimental results, the power consumption due to the proposed architecture is reduced by 67.58 percent when compared with that of the mesh architecture and 58.1 percent when compared with that of the WNoC architecture. This is because the proposed architecture does not use as many cores as other architectures do. The power consumption for transmission in the directory based wireless architecture is lower (than that of the WNoC architecture), because in the proposed architecture, the data is not sent to all other subnets individually. It should be noted that the amount of power consumption should be increased with the increased number of subnets.

Scalability is a growing concern, especially when the total number of cores increases. As the total number of cores increases, the total number of entries in the directory should increase. The proposed directory is dynamic in nature. Assuming that there is enough storage in the directory, the proposed directory helps make the system scalable by allowing more entries in the directory.

TABLE 8
Total Power Consumption for Three Architectures

Different Scenarios	Mesh Architecture	WNoC Architecture	Proposed Architecture
Case 1: (0,0.0)-(1,1.8)	Ptot = P1 + P2+ P3 P1 = (Pwr*Nwr) + (Pcwr*Ncwr) = 43 P2 = (Pwr*Nwr) + (Pcwr*Ncwr) = 43 P3 = Palwr+ Pcanw = 5.5 + 19.5 = 25 Ptot = 43+43+25 = 111	Ptot = Psd + Pds Psd = Pawrsn+(Pcwr*Ncwr) +Pcwl+3(Pwl+Pcasn) = 2.5+(3*2)+3.3+25.8 = 37.6 Pds = Pdsn+Pwl+Pssn = 11.8+1.1+11.8 = 24.7 Ptot = 37.6+24.7 = 62.3	Ptot = Psdr + Pcdr Psdr = Pawrsn+(Pcwr*Ncwr)+Pcwl+Pwl = 2.5+(3*2)+3.3+1.1 = 12.9 Pcdr = Pdr+Pcwl = 6 + 3.3 = 9.3 Ptot = 12.9+9.3 = 22.2
Case 2: (0,0.4)-(1,1.4)	P1 = 24, P2 = 24, P3 = 25 Ptot = 73	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 6.9, Pcdr = 9.3 Ptot = 16.2
Case 3: (0,0.7)-(1,0.1)	P1 = 23, P2 = 23, P3 = 25 Ptot = 71	Psd = 31.6, Pds = 12.7 Ptot = 44.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 4: (0,0.3)-(0,1.5)	P1 = 23, P2 = 23, P3 = 25 Ptot = 71	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 5: (1,0.5)-(0,1.2)	P1 = 23, P2 = 23, P3 = 25 Ptot = 71	Psd = 34.6, Pds = 21.7 Ptot = 56.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 6: (1,0.7)-(0,1.5)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 7: (0,1.0)-(1,0.0)	P1 = 27, P2 = 27, P3 = 25 Ptot = 79	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 8: (0,0.8)-(1,1.6)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 9:(0,0.7)-(0,1.1)	P1 = 7, P2 = 7, P3 = 25 Ptot = 39	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 10: (1,1.5)-(1,0.2)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 34.6, Pds = 21.7 Ptot = 56.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 11: (0,1.3)-(0,0.1)	P1 = 23, P2 = 23, P3 = 25 Ptot = 71	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 12: (0,1.4)-(1,0.6)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 31.6, Pds = 18.7 Ptot = 50.3	Psdr = 6.9, Pcdr = 9.3 Ptot = 16.2
Case 13: (1,0.1)-(1,1.1)	P1 = 15, P2 = 15, P3 = 25 Ptot = 55	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 14: (1,1.2)-(0,0.8)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 15: (1,0.6)-(0,1.2)	P1 = 11, P2 = 11, P3 = 25 Ptot = 47	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 16: (1,0.4)-(0,1.7)	P1 = 31, P2 = 31, P3 = 25 Ptot = 87	Psd = 31.6, Pds = 15.7 Ptot = 47.3	Psdr = 6.9, Pcdr = 9.3 Ptot = 16.2
Case 17: (1,1.3)-(0,1.3)	P1 = 15, P2 = 15, P3 = 25 Ptot = 55	Psd = 34.6, Pds = 18.7 Ptot = 53.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 18: (0,1.2)-(1,1.0)	P1 = 7, P2 = 7, P3 = 25 Ptot = 39	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 19: (0,0.1)-(1,0.7)	P1 = 23, P2 = 23, P3 = 25 Ptot = 71	Psd = 34.6, Pds = 15.7 Ptot = 50.3	Psdr = 9.9, Pcdr = 9.3 Ptot = 19.2
Case 20: (1,0.2)-(0,1.6)	P1 = 43, P2 = 43, P3 = 25 Ptot = 111	Psd = 37.6, Pds = 24.7 Ptot = 62.3	Psdr = 12.9, Pcdr = 9.3 Ptot = 22.2
Case 21: (0,0.6)-(0,0.5)	P1 = 15, P2 = 15, P3 = 25 Ptot = 55	Psd = 14.5, Pds = 14.5 Ptot = 29	Psdr = 14.5, Pcdr = 14.5 Ptot = 29
Case 22: (1,0.7)-(1,0.8)	P1 = 7, P2 = 7, P3 = 25 Ptot = 39	Psd = 8.5, Pds = 8.5 Ptot = 17	Psdr = 8.5, Pcdr = 8.5 Ptot = 17
Case 23: (0,1.4)-(0,1.2)	P1 = 11, P2 = 11, P3 = 25 Ptot = 47	Psd = 11.8, Pds = 11.8 Ptot = 23.6	Psdr = 11.8, Pcdr = 11.8 Ptot = 23.6
Case 24: (1,1.6)-(1,1.2)	P1 = 19, P2 = 19, P3 = 25 Ptot = 63	Psd = 17.5, Pds = 17.5 Ptot = 35	Psdr = 17.5, Pcdr = 17.5 Ptot = 35
Case 25: (0,1.7)-(0,1.1)	P1 = 11, P2 = 11, P3 = 25 Ptot = 47	Psd = 11.8, Pds = 11.8 Ptot = 23.6	Psdr = 11.8, Pcdr = 11.8 Ptot = 23.6

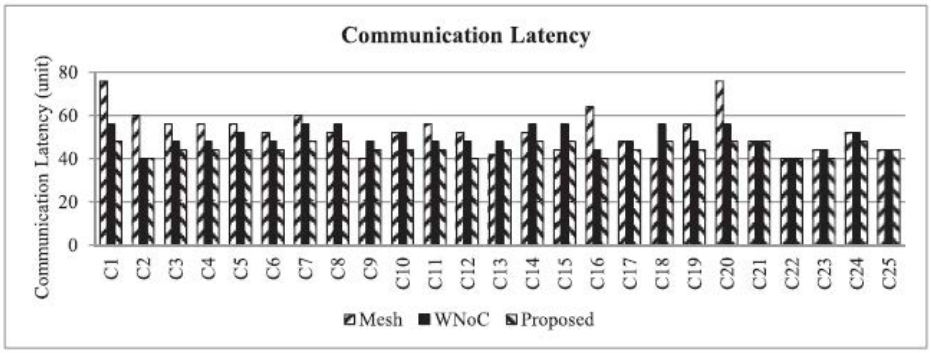

Fig. 8. Communication latency for different architectures.

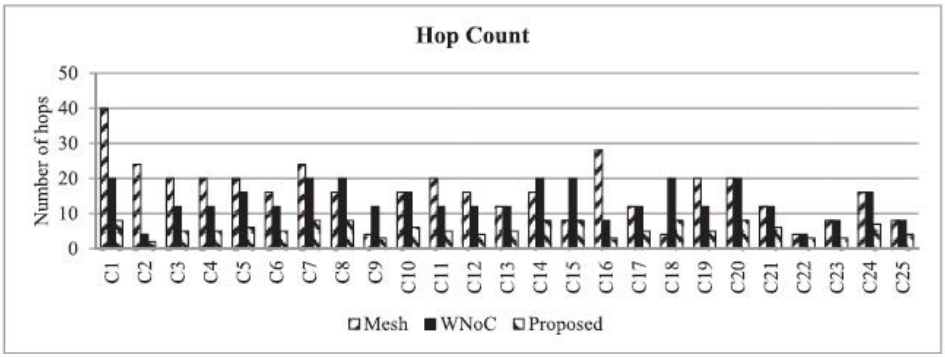

Fig. 9. Hop count for different architectures.

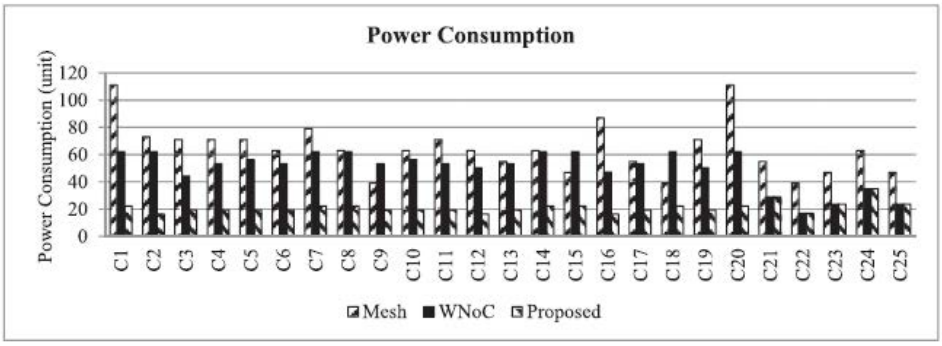

Fig. 10. Power consumption for different architectures.

CONCLUSIONS AND FUTURE WORK

Software developers generate more threads to take advantage of the multicore architecture. Multicore architectures help improve performance to power ratio by concurrently using multiple cores at a lower frequency. Contemporary multicore architectures have multilevel cache memory organization; each core has its own FLC (first level cache) and the system should have multiple levels of caches. Due to the presence of caches, multicore architectures suffer from high core-to-core communication latency (due to caches' dynamic behavior) and power consumption (because cache is power hungry). Studies suggest that directory based architecture with wire-less routers has potential to decrease communication latency

in multicore architectures. In this paper, we present a novel directory based mechanism to minimize communication latency in a multicore architecture by using wireless routers. We simulate popular 2D mesh, recently introduced WNoC, and the proposed directory based architecture. According to the experimental results, the proposed architecture helps decrease the communication delay by up to 15.71 percent and the total power consumption by up to 67.58 percent when compared with the mesh architecture. Similarly, the proposed architecture helps decrease the communication delay by up to 10.00 percent and the total power consumption by up to 58.10 percent when compared with the original WNoC architecture. This is due to the fact that the proposed directory based architecture helps reduce the total number of hops.

For future research, we plan to model and simulate multicore systems with distributed directories by applying various parallel techniques (such as data-level-parallelism and memory-level-parallelism) to study the impact on scalability, latency, and power consumption.

REFERENCES

[1] L. A. Barroso, et al., "Piranha: A scalable architecture based on single-chip multiprocessing," presented at the *27th Int. Symp. Computer Architecture, Vancouver*, BC, Canada, pp. 282–293, Jun. 2000.

[2] "From a few cores to many: A Terascale computing research overview," Intel, 2006. [online]. Available: http://download.intel. com/research/platform/terascale/terascale Accessed on: 2016

[3] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded Sparc processor," *Proc. IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar. 2005.

[4] K. Sankaralingam, et al., "Exploiting ILP, TLP, and DLP using polymorphism in the TRIPS architecture," in Proc. 30th *Annu. Int. Symp. Comput. Archit*., Jun. 2003, pp. 422–433.

[5] N. D. E. Jerger, "Chip multiprocessor coherence and interconnect system design," Ph.D. dissertation Dept. Elect. Eng., Univ. Wisconsin-Madison, Madison, WI, USA, 2008.

[6] D. K. Every, "IBM's cell processor: The next generation of computing?," *Shareware Press*, 2005. [Online]. Available: http://www. mymac.com/fileupload/CellProcessor.pdf, Accessed on: 2016.

[7] J. M. Li, P. Jiao, and C. G. Men, "The heterogeneous architecture of multicore research and design," in Proc. Int. Conf. Manage. Serv. Sci., 2009, pp. 1–6.

[8] H. V. Caprita and M. Popa, "Design methods of multithreaded architectures for multicore microcontrollers," in *Proc. IEEE Int. Symp. Appl. Comput. Intell. Inform.*, 2011., pp. 427–432.

[9] D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous multi-threading: Maximizing on-chip parallelism," in *Proc 22nd Annu. Int. Symp. Comput. Archit.*, 1995, pp. 392–403.

[10] S. Bell, et al., "Tile64-processor: A 64-core soc with mesh interconnect," in *Proc. IEEE Int. Conf. Solid-State Circuits*, 2008, pp. 588–598.

[11] S. Vangal, et al., "An 80-tile 1.28 TFLOPS network-on-chip in 65nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, 2007, pp. 98–99.

[12] "Quadro FX 5600," NVIDIA, [Online]. Available: http://www. nvidia.com, accessed on: 2016

[13] M. J. Saikia and R. Kanhirodan, "High performance single and multi-GPU acceleration for diffuse optical tomography," in *Proc. Int. Conf. Contemp. Comput. Inform.*, 2014, pp. 1320–1323.

[14] O. Villa, D. P. Scarpazza, and F. Petrini, "Accelerating real-time string searching with multicore processors," *IEEE Computer*, vol. 41, no. 4, pp. 42–50, Apr. 2008.

[15] K. Meng, F. Huebbers, R. Joseph, and Y. Ismail, "Modeling and characterizing power variability in multicore architectures," in *Proc. Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2007, pp. 146–157.

[16] R. E. Ahmed and M. K. Dhodhi, "Directory-based cache coherence protocol for power-aware chip-multiprocessors," in *Proc. 24th Canadian Conf. Electr. Comput. Eng.*, 2011, pp. 1036–1039.

[17] R. Cole and V. Ramachandran, "Efficient resource oblivious algorithms for multicores with false sharing," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp.*, 2012, pp. 201–214.

[18] Y. Ju and M. Park, “Performance improvements using application hints on a multicore embedded system,” in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. New York, NY, USA: Springer, 2013, pp. 161–172.

[19] B. Kordic, V. Marinkovic, M. Popovic, and V. Pekovic, “Parallel processing of multichannel video based on multicore architecture,” in *Proc. East. Eur. Reg. Conf. Eng. Comput. Based Syst.*, 2013, pp. 157–160

[20] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, et al., “Interconnect-aware coherence protocols for chip multi-processors,” in Proc. ACM SIGARCH Comput. Archit. News, vol. 34, no. 2, 2006, pp. 339–351.

[21] R. Jeyapaul, F. Hong, A. Rhisheekesan, A. Shrivastava, K. Lee, “UnSync-CMP: Multicore CMP architecture for energy-efficient soft-error reliability,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 254–263, Jan. 2014.

[22] S.-B. Lee, et al., “A scalable micro wireless interconnect structure for CMPs,” in *Proc. ACM. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 217–228.

[23] A. Shacham, K. Bergman, and L. P. Carloni, “Photonic networks-on-chip for future generations of chip multiprocessors,” *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1246–1260, Sep. 2008.

[24] D. Zhao and Y. Wang, “SD-MAC: Design and synthesis of a hard-ware-efficient collision-free QoS-aware MAC protocol for wireless network-on-chip,” *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1230–1245, Sep. 2008.

[25] M. F. Chang, et al., “CMP network-on-chip overlaid with multi-band RF-interconnect,” in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, 2008, pp. 191–202.

[26] S. Loucif, M. Ould-Khaoua, and L. M. Mackenzie, “On the performance merits of bypass channels in hypermeshes and k-aryn- cubes,” Comput. J., vol. 42, no. 1, pp. 62–72, 1999.

[27] W. J. Dally, “Express cubes: Improving the performance of k-ary n- cube interconnection networks,” *IEEE Trans. Comput.,* vol. 40, no. 9, pp. 1016–1023, Sep. 1991.

[28] C. Wang, W. H. Hu, and N. Bagherzadeh, “A wireless network-on-chip design for multicore platforms,” in *Proc. 19th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process*., 2011, pp. 409–416.

[29] J. H. Bahn, S. E. Lee, and N. Bagherzadeh, “On design and analysis of a feasible network-on-chip (NoC) architecture,” in *Proc. 4th Int. Conf. Inf. Technol., Inf. Technol. New Generations*, 2007, pp. 1033–1038.

[30] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann, 2003.

[31] H. C. Freitas, T. G. S. Santos, and P. O. A. Navaux, “Design of programmable NoC router architecture on FPGA for multi-cluster NoCs,” *Electron. Lett.,* vol. 44, no. 16, pp. 969–971, 2008.

[32] V. F. Pavlidis and E. G. Friedman, “3-D topologies for networks-on-chip,” *IEEE Trans. Very Large Scale Integration Syst.*, vol. 15, no. 10, pp. 1081–1090, Oct. 2007.

[33] H. Xiao, T. Isshiki, H. Kunieda, Y. Nakase, S.Kimura, “Hybrid shared-memory and message-passing multiprocessor system-on-chip for UWB MAC,” in *Proc. IEEE Int. Conf. Consum. Electron.*, 2012, pp. 658–659.

[34] W. J. Dally, “Wire efficient VLSI multiprocessor communication networks,” in *Proc. Stanford Conf. Adv. Res. Very Large-Scale Integr.*, 1987, pp. 391–415.

[35] D. Lenoski, et al., “The stanford dash multiprocessor,” *J. Comput*., vol. 25, no. 3, pp. 63–79, 1992.

[36] N. Blachford, Cell Architecture Explained Version 2 2016. [Online]. Available: http://www.blachford.info/computer/Cell/Cell0_v2.html, Accessed on: 2016

[37] A. Clements, *Computer Organization & Architecture: Themes and Variations*, 1st ed. CL Engineering, Stamford, CT, USA, 2013.

[38] D. M. Tullsen, S. J. Eggers, and H. M. Levy, “Simultaneous multi-threading: Maximizing on-chip parallelism,” *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 392–403, 1995.

[39] “VisualSim Architect,” Mirabilis Design, 2016. [Online]. Available: http://mirabilisdesign.com/new/visualsim/, Accessed on: 2016.

[40] “What consumes more power: Wired or wireless internet connection?,” *Comput. Netw.,* 2012. [Online]. Available: http://forum. allaboutcircuits.com/threads/what-consumes-more-power-wired-or-wireless-internet-connection.69734/

[41] R. Hui, “Comparison of power savings based on the use of wire-less charging systems & conventional wired power adapters,” *City Univ. of Hong Kong*, 2009. [Online]. Available: https://www. wirelesspowerconsortium.com/technology/comparison-of-power-savings.html