Studies on the Impact of Cache Configuration on Multicore Processor

Ram Prasad Mohanty

(Roll No: 611CS103)

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Odisha, India

May, 2014.

Studies on the Impact of Cache Configuration on Multicore Processor

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology

(By Research)

in

Computer Science and Engineering

by

Ram Prasad Mohanty

(Roll No: 611CS103)

under the guidance of

Dr. Ashok Kumar Turuk

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Odisha, India

May, 2014.

Dedicated to my parents and teachers

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that the work in the thesis entitled ***” Studies on the Impact of Cache Configuration on Multicore Processor”*** submitted by ***Ram Prasad Mohanty*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology (By Research) in Computer Science and Engineering, National Institute of Technology, Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Ashok Kumar Turuk
Associate Professor
Department of CSE
National Institute of Technology
Rourkela - 769008

Place: NIT, Rourkela

Acknowledgment

First of all, I would like to express my deep sense of respect and gratitude towards my supervisor Prof. Ashok Kumar Turuk, who has been the guiding force behind this work. I want to thank him for introducing me to the field of Multicore Technology and giving me the opportunity to work under him. His undivided faith in this topic and ability to bring out the best of analytical and practical skills in people has been invaluable in tough periods. Without his invaluable advice and assistance it would not have been possible for me to complete this thesis. I am greatly indebted to him for his constant encouragement and invaluable advice in every aspect of my academic life. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

Secondly, I would like to thank Prof. Bibhudatta Sahoo for his invaluable suggestions, and encouragements during this research period. I also thank Prof. Santanu Kumar Rath, Prof. Pabitra Mohan Khilar and Prof. Santanu Kumar Behera for serving on my Masters Scrutiny Committee.

I wish to thank all faculty members and secretarial staff of the CSE Department for their sympathetic cooperation.

During my studies at N.I.T. Rourkela, I made many friends. I would like to thank them all, for all the great moments I had with them.

When I look back at my accomplishments in life, I can see a clear trace of my family's concerns and devotion everywhere. My dearest mother, whom I owe everything I have achieved and whatever I have become; my beloved late father, for always blessing me and inspiring me to dream big even at the toughest moments of my life; and my brother and sister; who were always my silent support during all the hardships of this endeavor and beyond.

Ram Prasad Mohanty

Abstract

The demand for a powerful memory subsystem is increasing with increase in the number of cores in a multicore processor. The technology adapted to meet the above demands are: increasing the cache size, increasing the number of levels of caches and by means of a powerful interconnection network. Caches feeds the processing element at a faster rate. They also provide high bandwidth local memory to work with. In this research, an attempt has been made to analyze the impact of cache size on performance of multicore processors by varying L1 and L2 cache size on the multicore processor with internal network (MPIN), also referenced from NIAGRA architecture.

As the number of cores increases, traditional on-chip interconnect like bus and crossbar proves to be less efficient as well as suffers from poor scalability. In order to overcome the scalability and efficiency issues in these conventional interconnects, ring based design has been proposed. The effect of interconnect on the performance of multicore processors has been analyzed and a novel scalable on-chip interconnection mechanism (INoC) for multicore processors has been proposed. The benchmark results are presented using a full system simulator. Results shows that, using the proposed INoC, execution time can be significantly reduced, compared with MPIN.

Cache size and set-associativity are the features on which the cache performance is dependent. If the cache size is doubled, then the cache performance can increase but at the cost of high hardware, larger area and more power consumption. Moreover, considering the small form-factor of the mobile processors, increase in cache size affects the device size and battery running time. Re-organization and reanalysis of cache configuration of mobile processors are required for achieving better cache performance, lower power consumption and chip area. With identical cache size, performance gained can be obtained from a novel cache mechanism. For simulation, we used SPLASH2 benchmark suite.

Dissemination of Work

Published:

1. **Ram Prasad Mohanty**, Ashok Kumar Turuk and Bibhudatta Sahoo, ”Analysing the Performance of Divide and Conquer Algorithm on Multi-core Architecture”, International Conference On Computing, Communication and Sensor Networks (CCSN 2012), pp. 7-11, Rourkela, 2012.

2. **Ram Prasad Mohanty**, Ashok Kumar Turuk and Bibhudatta Sahoo, ”Performance Evaluation of Multi-core processors with Varied Interconnect Networks”, International Conference on Advanced Computing, Networking and Security (ADCONS 2013), pp. 7-11, Mangalore, 2013, DOI: 10.1109/ADCONS.2013.40.

3. **Ram Prasad Mohanty**, Ashok Kumar Turuk and Bibhudatta Sahoo, ”Modeling and Analyzing Cache for Multi-Core Processor”,CiiT International Journal of Programmable Device Circuits and Systems, pp. 79-85, volume 6, Number 3, March 2014.

4. **Ram Prasad Mohanty**, Ashok Kumar Turuk and Bibhudatta Sahoo, ”Architecting Fabricated Implant in Interconnect for Multi-core Processor”, In Advanced Computing, Networking and Informatics-Volume 2, pp. 517-524, Springer International Publishing, 2014.

Contents

List of Figures

List of Tables

List of Acronyms

ILP	Instruction Level Parallelism
CMP	Chip Multi Processors
ISA	Instruction Set Architecture
SMT	Simultaneous Multi-Threading
KB	Kilo Byte
MB	Mega Byte
MPIN	Multi-core Processor with Internal Network
INoC	Interconnection Network on Chip
FFT	Fast Fourier Transform
MID	Mobile Internet Devices
CPU	Central Processing Unit
GPU	Graphical Processing Unit
TLP	Thread Level Parallelism
LRU	Least Recently Used
FIFO	First In First Out
L2	Level 2 cache
L1	Level 1 cache
DDR	Double Data Rate
DIMM	Dual In-line Memory Module

List of Symbols

$index$	Cache index value
F_e	Fraction Enhanced
S_e	Speedup enhanced
$*$	Multiplication
$+$	Addition
$-$	Subtraction

Chapter 1

Introduction

The quest for enhancing computational power is never-ending. A few of the techniques for enhancing processors computational power are: exploiting parallelism, increasing number of cores in a processor chip etc. Innovation in software is also desirable so that, the software can effectively use the available cores. For years, leading processor manufacturers like Intel have focused on increasing the number of pipeline stages in order to enhance processor clock speed. They have exploited parallelism among instructions, to achieve processor speed. However, there is an inherent limitations on the degree of parallelism that can be exploited among instructions. A single core processor, as shown in Figure 1.1 [27] attempts to improve either the clock speed or the number of instruction executed per clock. The above is achieved by exploiting instruction level parallelism. This limitation compelled the leading chip manufacturers towards *multicore*, and *multi-thread* technology. Multicore processors are also known as Chip Multi-Processors (CMP) [62], in which multiple cores are integrated on a single chip. A few of the general purpose multicore processors are Core 2, and Nehalem from Intel; Athlon, and Phenom from AMD. Apart from general purpose processor, a few multicore embedded processor such as Cortex-A9, ARM11 MPCore from ARM are also available. A multicore processor is capable of executing multi-threaded applications faster, compared with multiprocessor system consisting of multiple single cores. Because, of shorter distance between cores in a multicore processor, the communication among the cores are faster. It is also cheaper to have multiple cores on a single die than multiple single core coupled together as shown in Figure 1.2 [86] . Figure 1.1 represents a single core processor, Figure 1.2 represents a multiprocessor in which two different processors are combined as one. Figure 1.3 represents a multicore processor with private cache which has multiple processing cores within the same processor die and each core having its own private cache. Figure 1.4 represents a multicore processor with shared cache in which all cores share the same cache.

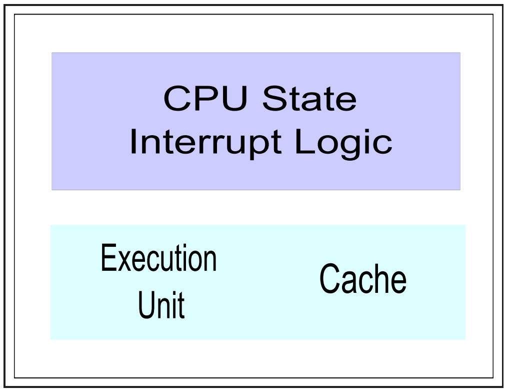

Figure 1.1: Single Core Processor

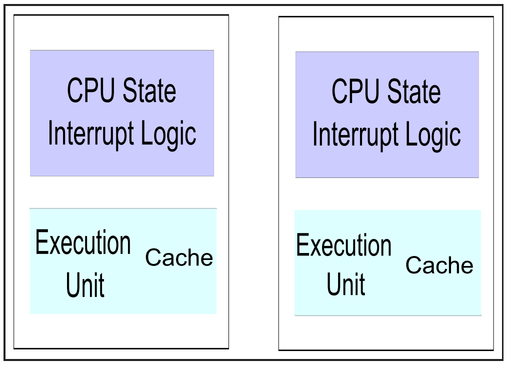

Figure 1.2: Multiprocessor Architecture

1.1 Multicore Processor

Multicore technology have evolved overtime. This technology has become the mainstream of commercial chip manufacturers like Intel and Advanced Micro Devices (AMD) [71]. Multicore processors are considered as an immediate solution to the current challenges in processor design. It has the capacity and capability of executing applications more efficiently compared with single core processor. Multicores has the efficiency of executing not only current applications but future complex applications as well. A multicore processor comprises of more than one independent cores. These cores are capable of running multiple instructions at the same time. They are integrated on a single die or onto multiple dies on a single chip package. In conventional multicore processors, the instruction set architecture (ISA) for each core is similar with that of uniprocessor, with minor modification to support parallelism [8].

Most of the multicore processors have one or more levels of memory hierarchy within the processor cores as shown in Figure 1.3. While some share the memory interface, others share the cache levels as shown in Figure 1.4. The design of shared resource is advantageous when there is an imbalance of resource demand between different cores. In this design, one core can potentially use the entire cache space resulting in higher resource utilization. Multicore processors have improved performance, power consumption as well as thermal effect compared to single core processors. Multiple cores run parallel at lower clock speed resulting in lower power consumption. But, as the enhancement in speed and number of transistors per core is increased it is difficult

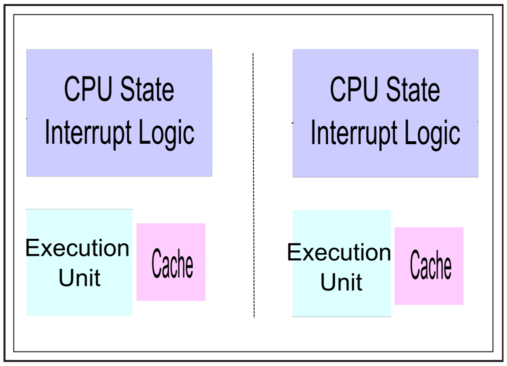

Figure 1.3: Multicore Processor Architecture

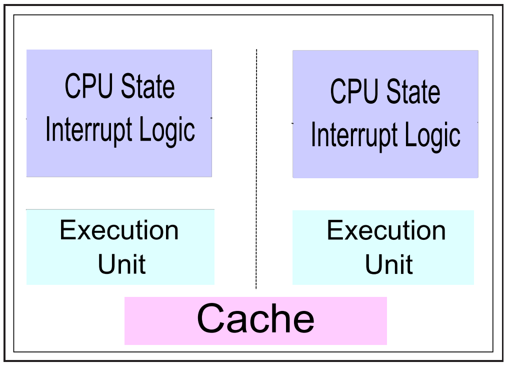

Figure 1.4: Multicore Processor with Shared Cache Architecture

to keep pace using symmetric multicores. More performance enhancement can be made if non-uniform cores with varied characteristics and capability can be packed into a single die. This lead to the development of asymmetric multicore processor also known as heterogeneous multicore processor. In heterogeneous multicore processors cores are with different architectural, functional and performance characteristics [33]. Difference in the execution bandwidth or super-scalar width, in-order, out-of-order and cache size are a few of the characteristics that make these core heterogeneous [54].

1.2 Design Issues with Multicore Processors

Introduction of multicore processor has considerably enhanced the performance with respect to single core processors [54]. However, putting multiple cores on a single chip has led to new problems. It is observed that heat dissipation and power consumption have increased exponentially. For proper functioning of multicore processor, this issue must be addressed. Another issue that occurs in the multicore is the memory inconsistency. Moreover, there will be no benefit if the programmers do not develop applications to take the advantage of multicore. Performance enhancement is possible through innovative organization of multicore processor. A few of the issues associated with multicore processor are described below:

- ***Enhancement in the memory system:*** It is essential to increase the memory size when number of cores are placed on a single chip. Increasing the memory size does not guarantee the improvement of memory subsystem. A higher size memory as well as proper cache configuration will enhance the performance of memory subsystem. Performance of an application is dependent on the number of memory stall cycles, which in turn is dependent on *miss rate*, *miss penalty* and *memory accesses per instruction*. *Miss rate* is an important measure of cache design. *Miss rate* varies with cache size [39]. Cache enables a faster access to data. Researchers are more inclined towards analyzing the impact of cache performance on execution time in a multicore environment. Peng et al. [66] have studied the impact of memory on performance and scalability of dual core processors from leading manufacturers like *Intel* and *AMD*. In their research work it was established that the memory subsystem has higher impact on the performance of multicore processors. Julian et al. [13] have established the relationship of memory system on performance in single as well as multicore processors. Through simulation they showed that memory latencies as well as communication overhead are prominent factors that limit the performance of multicore systems. They have also shown that a suitable cache configuration can enhance the performance by reducing the cache coherence and communication latencies. Performance of a multicore processor is also influenced by the cache access time [52].

- ***Enhancement in the Interconnection Networks:*** Interconnection network establishes communication between processor cores and memory sub-system in multicore processors. As the main memory size gets enlarged, it necessitates a proper communication management to handle memory requests. The interconnection network that exists between the cores has become a prime concern for the manufacturers [19]. When a network gets faster, the communication latency between cores as well as memory transactions is reduced. Wang et al. [82] have made a quantitative analysis to study the impact of *on-chip network* and *memory hierarchy* on the performance of multicore processors. They studied the memory hierarchy design for multicore processors, and their performance by varying memory hierarchy. The effect of on-chip network on the performance of multicore processor is also studied. William J. Daily [19] has analyzed the performance of k-ary, n-cube interconnection network. It is shown that low-dimensional network has higher hot-spot throughput as well as lower latency for the same bisection bandwidth. Matteo Monchiero [60] has presented a detailed evaluation of multicore architectures. The architecture considered for evaluation comprises of private L1 cache, shared or private L2 cache connected through a shared bus interconnect with multiple cores. They also explored design space, considering number of cores, processor complexity as well as L2 cache size. They detailed the effect of these configurations on energy consumption and performance in addition to temperature. They have also analyzed the effect of chip floorplan on its thermal behavior. Different placement of L2 cache can lead to variation in the hotspot of a die. A floor plan in which the processor is surrounded by L2 cache can be cooler by 0.5°C.

More, recently *Intel* has come up with Quick Path Interconnect, to provide high-speed point-to-point link on both sides of the processor. The speed of transfer is enhanced because of a connection between the distributed shared memory, I/O hub, Intel processors as well as internal cores. For high-speed data transfer AMD developed a hyper transport technology which is a wide bus based system. Similarly, a new interconnect is seen in the *TILE64 iMesh*. This mesh consists of five networks for high interaction between the I/O and the off-chip memory. But, till date the question remains open as which type of communication yields the most optimized result for multicore processors [55].

- ***Handling of Power and Temperature*: ** Heat dissipation increases with increase in power consumption. Putting more number of cores on a single chip not only increases the power consumption but also the amount of heat generated. This may lead to combustion of computer in extreme cases. In order to avoid such cases, individual cores are executed at lower frequencies. The provision to shut down the unutilized cores is built into the system, so as to restrict the power consumption. Heat generation is taken care by restricting the number of hot spots over the chip. This is handled at the design level. The design is chosen such that the hot spots does not grow as well as the heat generated is uniformly spread across the chip. Power and temperature management is considered as the first-level constraints in the design process of a processor [60, 71]. D. Geer [30] have made a detailed analysis on the advantages as well as issues with a few multicore processors of leading providers like Intel, AMD, Sun, etc. Evaluation of design space for multicore architectures to analyze the power consumption and heat dissipation has been made by M. Monchiero et al. [59]. Parameters like cache size, interconnect, floorplan not only affects the performance of multicore processor but also their power consumption and heat dissipation.

- ***Cache Coherence:*** Distribution of L1 and L2 caches across the chip is another area of concern in a multicore environment [71]. When each core has its own individual cache then data at each cache might not hold the most recent values or the actual required values [69]. Two types of protocols are used in general for handling cache coherence. They are Snooping protocol and Directory based protocol. Snoopy protocol is based on two states. Using these states it can determine which values in the cache needs to be updated. The snooping protocol is not scalable. Directory based protocol is scalable, and can be used on an arbitrary network. Thus it can be adapted to multiple processors [86]. Hackenberg et al. [37] have made a comparative analysis of various cache architectures in addition to coherency protocols. They have made the analysis on x86-64 multicore SMP systems and concluded that memory subsystem and cache coherence protocol governs the performance of multicore processor.

  Cheng et al. [16] have proposed an adaptive cache coherence protocol optimized for producer consumer sharing. They have analyzed the impact of cache coherence on performance of multicore processor. Author's have suggested an innovative coherence protocols for emerging multicore processors. Khan et al. [49] have proposed a *dependable cache coherence multicore architecture*. Their proposed coherence protocol is based on the traditional directory based protocol, and a novel execution-migration-based architecture, which is transparent to the programmer. This architecture allows only a single copy of data to be cached across the processor. Whenever a thread access an address which is not available on the local cache of the core where it is executing on, then the thread migrates to the appropriate core where the address is present and continues its execution in that core. Brown et al. [11] have proposed a proximity-aware directory based protocol for handling cache coherence in multicore processor.

- ***Multi-threading and Parallel Programming:*** To take advantage of multiple cores, the concept of multi-threading has to be built into the application program. Reconstructing applications to embed multi-threading makes the programmers revise the application in most of the cases [18]. Applications need to be written by the programmers to make the subroutine capable of executing on various cores. This requires to handle data dependence in a synchronized and structured way. Applications are not effectively using the advantages of multicore system, if a particular core is used more than the other. A few companies have manufactured their product with the capability of utilizing advantage of multicore [31]. The recent release of applications from Microsoft, and Apple can execute in four cores [31].

Programmers must write the codes so that they can be divided and executed in parallel on multiple cores.

- ***Cores are not getting the Data:*** In multicore environment, one or more cores may remain idle, waiting for the data. This will occur if a program is not efficiently developed to utilize the cores of a multiprocessor. This is visible if the multicore system is used to run a single-threaded application. In such case thread will run on a single core while other cores will remain idle. Though the cores, remined idle, yet they make calls to the main memory, wasting a lot of clock cycle. This adds to the penalty, reducing overall performance [71].

- ***Homogeneous or Heterogeneous Cores:*** Whether to have homogeneous core or heterogeneous core in a multicore environment has been a debate amongst the architects. Most of the real life processors have homogeneous cores. That is they have the same cache size, operational frequency, functions , etc. Whereas, in case of a heterogeneous or asymmetric system each core can have different memory model, function, frequency. CELL processor is an example of a multicore processor that has heterogeneous environment. It has a single power processing element and eight other synergistic processing element [71, 33]. Asymmetric processors have better performance than the symmetric ones. It may take some time to have the mainstream processors available with heterogeneous cores.

The key advantage of asymmetric processor is that each core is specialized to accomplish a specific task. But, development of applications is more complex in comparison to that of symmetric processors.

1.3 Motivation

With increase in number of cores in a chip, there has been a singnificant increase in the demand for (i) power, (ii) memory, (iii) efficient communication network. As the power consumption of multicore processor is more, the heat dissipation from them is also more. To resolve the heat dissipation, cores are made to run at variable frequencies and the idle cores are also not down. The increasing demand for memory can be met by increasing the memory size and memory bandwidth. Increasing the memory size and bandwidth will significantly affect the chip area, temperature and power consumption. Moreover, an efficient interconnection network is required for communication between memory and cores. This significantly adds to the complexity and cost of processor design. Along with the memory system and interconnection network, it is also important for the designer to select whether to have homogeneous or heterogeneous core in the processor [22]. Another challenge to multicore design is the increasing gap between processor performance and memory performance. It is essential for a processor designer to consider the memory system, interconnection network in addition to the processor cores for enhancing the performance of a processor [25].

One of the major goal of any designer is to reduce the cache access time, which significantly affects processor performance. The current advancement in cache memory subsystem includes additional levels of cache and enhancement in cache size to enhance the processor performance.

On-chip interconnect is used as a communication subsystem in multicore processor. Interconnection among the cores in a multicore processor has posed a great challenge. With an increase in the number of cores, the traditional on-chip interconnect like bus, and crossbar has proved to be less efficient. Moreover, they also suffer from poor scalability [1]. Also with the steady growth in the number of cores, the ring based interconnect has become infeasible. This necessitates, the designer to look for a novel way of interconnect among the cores without degrading the efficiency and scalability [12].

Multicore mobile processors are featuring predominantly in end user computing devices like mobile, smart phones, etc. Unlike general-purpose processors, an increase in the number of cores in mobile processors with a smaller form-factor and without cooling fan, will lead to higher power consumption and heat dissipation. Performance of these single chip mobile processors can be enhanced by designing appropriate cache subsystem with lower power consumption and chip area [45, 70]. Reorganization and reanalysis of cache configuration of mobile processors is required for better cache performance with lower power consumption and chip area.

1.4 Objective

With the motivation as outlined in the previous section, the objectives of our research work are identified as follows:

- To analyze the effect of cache size on the performance of multicore processors using different performance metrics.
- To propose an optimal cache configuration for multicore processors.
- To propose a scalable interconnection mechanism for multicore processors, and compare with existing architecture.
- To propose an optimal cache configuration for enhancing the performance of multicore mobile processor.

1.5 Organization of the Thesis

Rest of the thesis is organized into the following chapters:

Chapter 2: A survey of cache, and interconnection network as reported in the literature is presented in this chapter.

Chapter 3: This chapter proposes a cache configuration for multicore processor. The effect of cache on the multicore processor is analyzed. The metrics considered for analyzing the performance are: block size, sets-associativity, L1 cache size, L2 cache size, execution time and speedup.

Chapter 4: A novel interconnect named Interconnection Network on Chip (INoC) for multicore processor to reduce the communication delay and address the scalability issue is proposed in this chapter. To evaluate the proposed interconnection network the following parameters are considered: core size, execution time, and speedup.

Chapter 5: A cache configuration for multicore processors used in mobile phones have been proposed. The parameters considered for evaluating the performance are block size, sets-associativity, and L2 cache size.

Chapter 6: A few conclusions, along with the future scope of research are mentioned in this chapter.

Chapter 2

Design Consideration for Multicore Processor

2.1 Introduction

Computational power gained from multicore processor is enormous compared to single core processor. Though the performance of multicore depends on the number of cores in a single chip, yet it is not directly proportional to the number of cores. There are a few issues that restricts the performance of multicore processors. Multicore performance issues are primarily concerned with the intra-core, inter-core communication with the rest of subsystems. Numerous approaches have been adapted to address the multicore design issues. A detailed of it is available in [3, 8]. Multicore processors are also built into hand held devices like mobile. But, the design considerations for multicore processors in hand held devices are different from general purpose processors. The form factor of mobile devices is smaller which necessitates innovative design to reduce the power consumption and heat dissipation [45]. Researchers have established cache size, set-associativity, connectivity, etc. as some of the major factors for design consideration of high-performance multicore processors [6, 18, 9, 12]. Execution time, speedup are being used by the researchers to evaluate the alternative design for multicore processors [13, 19, 29]. Innovative designs proposed for performance enhancement in multicore processors are evaluated by various Benchmarks like PARSEC, SPLASH2, etc, are used to evaluate the performance of multicore processor.

In this chapter, we made a detailed survey on communication delay, performance degradation vis-a-vis cache configuration in a multicore processor.

2.2 Approach for Communication

In general, communication in a multicore processor is broadly classified as inter-core and intra-core communication. Inter-core communication is the communication among the cores, whereas the intra-core is the communication within the core. Thread support available within a core corresponds to the resources attainable with that core. A single core may support one or multiple threads. Intra-core communication or local communication corresponds to the communication between the threads within a core. Inter-core communication or global communication corresponds to the communication between the cores in a processor. Different types of interconnects have been used to provide the connectivity within the multicore processor [14]. The knowledge of these communications are used by the programmer for effective utilization of multicore processor. As more cores are assembled in a single die, the inter-core communication costs increases with distance between the cores. This enhancement in the number of cores, necessitates the need for innovative interconnects [22].

2.2.1 Off Chip Connections

An optimized on-core communication architecture is less effective without sufficient memory bandwidth. Ideally, each core would have its own memory channel or a channel shared between small number of cores. Chip's packaging puts a constraint on the number of possible off-chip memory channels. With multiple cores, all with potentially different working sets, the problem gets worse. Recent developments in 3D integration technologies might overcome this problem to a significant level. Memory could be stacked directly on top of the cores in the same package; this could both increase the number of potential memory channels available as well as provide a reduction in memory access latency [14, 53].

2.2.2 Going beyond Wires

Wires are not the only way to communicate within a chip. Recently, the possibility of optical interconnects has been investigated, including numerous other exotic wireless communication techniques. Whilst these techniques may not overcome the communication problem in a bigger size core, they can be used for off chip connections. This will help in reducing the power required for driving external connections as well as decreasing the number of pins required per package, further allowing greater off chip bandwidth. New developments in photonics make it possible to produce photonic components in CMOS in a practical way, which enables both on and off chip optical communication [36]. An on-chip photonics can be used to construct an interconnection network potentially built out of a hybrid of electric and optical connections. While the 3D integration technology would allow a separate photonic plane to be constructed above the logic plane that performs a computation. One such potential design is proposed by Shacham et al. [72]. Compared with a conventional electrical design the hybrid of electric and optical connection reduce the network power consumption from around 100W to 4-5W for a large bandwidth [21].

2.2.3 Interconnect Architecture

Various types of interconnect exists for intra-chip communication like bus, ring, crossbar as well as network-on-chip (NoC). Each types have their own merits and demerits. The shared bus has long been used as a basic interconnection mechanism for both chip to chip communication, and inter-core communication. The major advantage of the shared bus lies in its simplicity, as everything that wishes to communicate on the bus simply shares wiring [8]. The shared bus is access by a device connected to the bus by means of a bus arbitrator. However, as the number of potential bus master grows the contention increases. This forces the bus either to handle significantly greater volumes of traffic or the bus master is forced to wait. Because of the above reason a single shared bus is a poor choice for multicore systems. Moreover it is difficult to build a single shared bus that runs over an entire chip, while maintaining reasonable bandwidth without consuming too much power [58]. The inability to communicate across a chip within a single cycle and the need to utilize the available wiring to maximize the communication efficiency led to the introduction of on-chip interconnection network or network-on-chip (NoC). While systems using many processing nodes that required an interconnection network have existed for many years [8, 64], but the development of NoC is relatively new concept [8].

2.3 Cache Organization for Multicore Processor

In an uniprocessor design, the memory sub-system is a component having a few levels of cache to provide the single processor with instructions and data. But, with the introduction of multicore processor caches have become a part of the memory system amongst the other components like intra-chip interconnect, in addition to cache coherence support. Increasing the number of cores per chip, demands a powerful memory subsystem [58]. Earlier versions of Intel Xeon processors establishes connection between processors and a single off-chip memory controller with aid of a front-side bus. In multiprocessor system this architecture limits the scalability across memory bandwidth in addition to interconnect bandwidth. This issue was by Intel with the introduction of Nehalem microarchitecture which has an on-chip memory controller to reduce the memory latency and increase the bandwidth scalability [37]. Multicore processors have enhanced the importance of cache. Cache provides processing elements a high bandwidth and faster local memory to work with [37]. Cache size can be enhanced for general purpose multicore processors but for mobile and other hand held devices, the cache size cannot be enhanced to a greater extent because of the size, power and heat dissipation in such devices. Such devices need a smaller size processor that consumes less power, and dissipating less heat. Cache size and set-associativity are the two features on which the cache performance is dependent on [44, 45]. If the cache size is doubled, then the cache performance can increase considerably but at the cost of more hardware, area and power consumption [8]. Re-organization and re-analysis of cache configuration for mobile processor is required for achieving better cache performance, lower power consumption, and lesser chip area [45]. Different approaches have been made by researchers to enhance the performance of multicore processors. One such approach is to alter the cache memory for multicore processors. Other approaches include: (i) addition in levels of cache such as L1, L2 L3, etc., (ii) alterations in cache classification having a unified cache or separate data and instruction cache, (iii) shared, or private cache, (iv) making updation in the locality of reference, (v) varying the cache policy [25, 87].

2.3.1 Cache Classification

Caches can be classified into the following:

- Instruction Cache: This cache holds only the instructions. Its design is optimized for instruction stream only.

- Data Cache: It holds only the data stream. The data caches reduces the latency of data fetch [6, 34].

- Unified Cache: This type of cache holds both instructions as well as data stream.

2.3.2 Cache Policy

Different cache policies also affects the performance of multicore processors. Cache policies can be categorized into following three types:

- Placement Policy: The mapping of addresses between the main memory and the cache is termed as placement policy.

- Fetch Policy: Fetch policy is an algorithm that determines the conditions on which a fetch is triggered from main memory to the cache. Also it specifies when the CPU can resume after a cache miss.

- Replacement Policy: Whenever there is a cache miss it may require to evict a block from the cache to main memory, to create a space in the cache for the new block to be fetched. Replacement policy also determines which block to be evicted.

2.4 Performance Metrics

This section describes a list of parameter which affects the performance of multicore processors. It also describes the metrics for performance evaluation of multicore processors.

Cache configuration is specified by number of sets, block size, associativity and memory bandwidth. Similarly an interconnection mechanism can be configured by describing its connectivity and bisection width.

Associativity: Cache associativity specifies how a cached data is associated with respect to locations in main memory. In a full associative cache, the cache controller can place a block from main memory at any location in cache memory.

Block Size: It specifies the amount of data that is transferred between cache and main memory on a cache miss.

Sets: This value provides the number of sets that can be allocated in a cache.

Latency: It is the time taken to transfer a block of data between main memory and caches.

Memory Bandwidth: It is the rate at which data can be read from or written into main memory. It is usually expressed in units of bytes/second.

Cache Hit: If a requested block is present in the cache, then we call it a cache hit.

Cache Miss: If a requested block is not present in the cache then we call a cache miss. When a cache miss occurs the required block has to be read from main memory to cache memory.

Access Time: The time required to transfer a word from cache memory to processor when it is a cache hit.

Miss Penalty: If requested block is not present in a cache, then the block needs to be brought from main memory. The time taken to bring a desired block into the cache is called miss penalty [44].

Misses per Instruction: Misses per instruction is expressed by the equation given below.

$$\frac{Misses}{Instruction} = \frac{Miss\,Rate * Memory\,Access}{Instruction\,Count} = Miss\,Rate * \frac{Memory\,Access}{Instruction}$$

Average Memory Access Time (AMAT): Average memory access time can be expressed by the equation given below [39].

$$AMAT = Hit\,Time + Miss\,Rate * Miss\,Penalty$$
$$AMAT = Hit\,Time_{L1} + Miss\,Rate_{L1} * (Hit\,Time_{L2} + Miss\,Rate_{L2} * Miss\,Penalty_{L2})$$

The expression shown below computes the index value of a cache when the cache size, block size and set associativity is given.

$$2^{index} = \frac{Cache\,Size}{Block\,Size * Set\,Associativity}$$

For a 64 KB , 2-way set associative cache with a 64 byte block size, and LRU replacement, the index can be given as:

$$2^{index} = \frac{64KB}{64 * 2} = 512 = 2^{9}$$

Thus the cache index is 9 bits.

Connectivity or degree: The number of edges/ links/ channels that are incident on a node is called the node degree.

Diameter: The diameter of a network is the longest distance between any two nodes in the network. A low diameter network is preferred because diameter puts a lower bound on complexity of parallel algorithm.

Bisection width: The minimum number of edges that must be removed in order to divide the network into two halves. Higher bisection width is preferred. This is because, in algorithms requiring large movement of data, the size of dataset divided by the bisection width puts a lower bound on the complexity of parallel algorithm [20].

CPI: The average number of clock cycles required for execution of each instruction.

Execution time: It is time required for the execution of a program. This specifies the amount of CPU time needed by the program. It is the product of clock cycle time and the number of clock cycles.

Speedup: Speedup is a metric for evaluating the performance alternative design. As stated byAmdahl, speedup is the ratio between the execution time of the old to the execution time of new, when some performance enhancement is made. The expression for computing speedup as given by Amdahl [39] is stated below:

$$Speedup = \frac{Performance\,for\,entire\,task\,using\,the\,enhancement\,when\,possible}{Performance\,for\,entire\,task\,without\,using\,enhancement}$$

$$Speedup_{overall} = \frac{Execution\,Time_{old}}{Execution\,Time_{new}} = \frac{1}{(1-F_e)+(\frac{F_e}{S_e})}$$

$$Execution\,Time_{new} = Execution\,Time_{old} * (1 - fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}$$

The above parameters and metrics are used to evaluate the performance of multicore processors as well as multicore mobile processors.

2.5 Simulation Platform and Benchmarks

In this section we describe, the simulation tool selected for performance comparison. In addition to the simulation platform, a set of benchmark program is used for evaluation is also described.

2.5.1 Simulation Platform

We have selected Multi2Sim for performance evaluation. Multi2Sim [80] is an open-source modular as well as fully configurable toolset, which facilitates the simulation of x86, ARM, MIPS, AMD, and NVIDIA architecture. For evaluation of any computing platform, it is essential to have an accurate and detailed simulation tool [78]. Wu et al. [85] have used Multi2Sim to create an simulation environment and implement a new cache management with Partitioning-Aware Eviction and Thread-Aware Insertion/ Promotion Policy. Antoni et al. [68] have simulated their Future kilo-x86-64 core processors with the aid of Multi2Sim. Kim et al. have made a comparative analysis of multi-thread system and chip multiprocessor system using Multi2Sim simulation framework [51]. We have used Multi2Sim to create a simulation environment that fits to our model with different memory, cores and communication configurations on x86 architecture. The referential architectural model used for simulation is depicted in the Figure 2.1. A set of application program is used as benchmark for the performance estimation and analysis of cache configuration for multicore processor, multicore mobile processor as well as interconnection mechanism is described in the next section.

2.5.2 SPLASH2 Benchmark

This section describes the benchmark program used for performance evaluation. SPLASH2 suite [83] enables the study of distributed as well as centralized shared address-space multicore processors. To study the impact of cache, on execution time, Subramanian et al. [69] have used benchmarks from SPLASH2 benchmark suite. Wang et al. [81] have made a real time cache performance evaluation for multicore processor using the benchmarks of SPLASH2 suite. Similarly various other researchers have also used this benchmark suite for carrying out their research in the area of multicore processors [38, 84]. This suite comes with benchmarks specific to applications as well as benchmarks specific to kernel. The SPLASH2 suite contains eight complete applications in addition to four computational kernels representing different scientific, graphics as well as engineering computations.. This suite has benchmarks specially designed for multicore processors.

For the performance analysis and comparison in our work, we have used Barnes, Cholesky and FFT application of this SPLASH2 suite [83]. The selected benchmarks comes under the application category. These benchmarks are selected because of their relevance to multicore processors and cache.

- FFT - The data set consists of the complex data points to be transformed, and another complex data points referred to as the roots of unity.

- Barnes -The Barnes application simulates interaction of a system of bodies (for example galaxies or particles) in three dimensions over a number of time-steps, using the Barnes-Hut hierarchical N-body method.

- Cholesky - The blocked sparse Cholesky factorization kernel factors a sparse matrix into the product of a lower triangular matrix and its transpose.

2.5.3 Multicore Processor with Internal Network (MPIN)

Figure 2.1 shows the architecture of a multicore processor with internal network. This architecture is primarily targeted for the commercial server class applications that have a low degree of Instruction Level Parallelism (ILP) but a high degree of Thread Level Parallelism (TLP). It has a private L1 cache dedicated for each core. L1 caches are connected to two L2 cache banks through a switch [78]. MPIN architecture uses a crossbar switch as interconnection network. A core can communicate with another core or L2 cache only through the switch. The crossbar allows up to eight simultaneous accesses of the L2 cache. Four memory controllers, interface L2 cache with main memory [14].

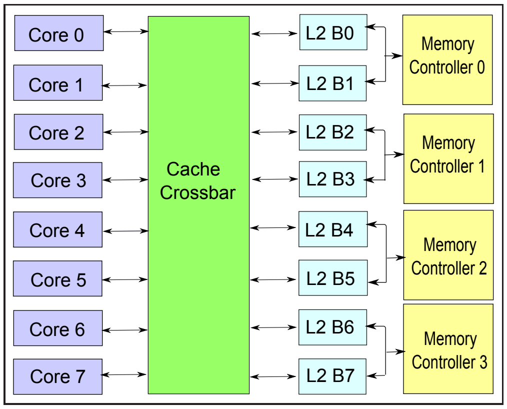

Figure 2.1: Multicore Processor with Internal Network

2.6 Summary

In this chapter, we briefly described the work relevant to multicore processors. It also gives a detailed view of metrics used for performance evaluation. A set of applications used as benchmarks for the performance estimation and analysis has been described. These applications are real-world examples and have different data access characteristics. FFT, Barnes and Cholesky benchmark are used for estimations and analysis as they are simple, and easier to find the number of cache line accesses during execution. Issues in multicore processors as well as multicore mobile processors have also been described. In the next chapter we studied the impact of cache configuration on multicore processor and proposed a suitable cache configuration for multicore processor.

Chapter 3

Cache Configuration for Multicore Processor

In this chapter, we made an attempt to analyze the impact of cache size on performance of multicore processors. We have also tried to estimate the performance of cache memory through parameters such as cache access time, miss rate, and also discussed the effect of cache parameters on execution time.

3.1 Introduction

The speed mismatch between processor and main memory has been widening. In multicore processor, this mismatch will be more, creating a bottleneck for memory access. An ideal memory access time should be one CPU cycle. To overcome the bottleneck memory organized into levels of hierarchy with the smaller and faster memory closer to the CPU. The memory hierarchy consists of registers, L1 cache, L2 cache, L3 cache, main memory, followed by secondary storage. Registers are the fastest memory elements and are closest to the CPU, followed by L1 cache. Secondary storage are far away from the CPU and have the highest memory access time.

Each level maps addresses from a larger memory to a smaller memory which is closer to the CPU. This concept is greatly aided by the principle of locality (both temporal and spatial) which refers to reuse of data and instructions by programs, which had been used recently. Figure 3.1 shows a typical system organization. In uniprocessor designs, memory system is a simple component, having a few levels of cache to provide data and instructions to the processor. In multicore processors

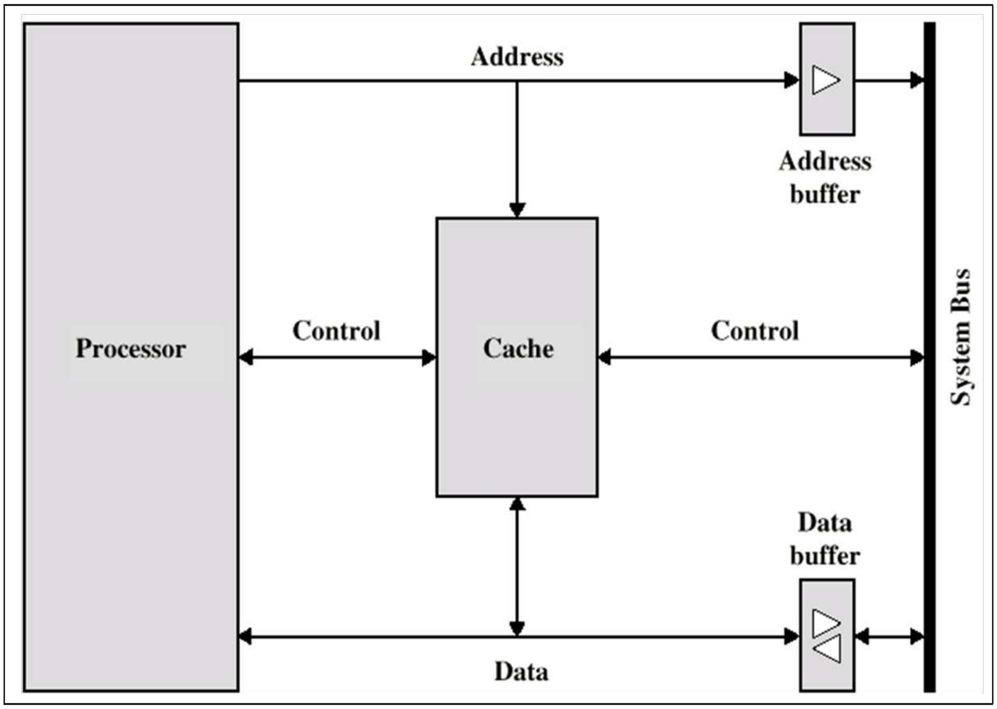

Figure 3.1: System Organization

caches have become a part of the memory system, the other parts being consistency model, intra-chip interconnect and support for cache coherence [8, 40]. Importance of cache configuration has increased in multicore processors. Caches have the option to be tagged and managed automatically by hardware or act as local memory, store. It is common to use automatic tagged caches. Amount of cache required depends on application running on the processor. From the performance point of view, it is preferred to have large size cache. But, it has been shown that large size caches are of little use for applications where requirements for data are less frequent [8]. Pawel et al. [32] have illustrated a few significant challenges which have come up with the introduction of multicore processors. They have presented a few evaluations with respect to industrial methodologies to handle the challenges. John Freuhe [28] have illustrated the importance of cache memory in multicore processor architectures, which will be designed to boost performance with minimal heat dissipation. Gal et al. [29] in their work have detailed the components that governs the performance of multicore processors. Authors have also outlined the importance of memory subsystem in the design of multicore processor in their work. Most of the researchers have used execution time as the metric for their analysis. They have studied the impact of memory hierarchy, and memory bandwidth on the execution time. In this work, we have analyzed the impact of cache size on the performance of multicore processor. The cache size of both L1 and L2 is varied.

3.2 Elements of Cache Design

The following features are taken into consideration, while designing a cache. They are Size, Mapping Function, Replacement Algorithm, Write Policy, Block Size and the number of caches [44, 39].

- Cache Size: Large cache can hold more of the program's useful data. Typically cache size is the product of three quantities. They are : (i) Number of sets, (ii) Block size, and (iii) Associativity. A cache with 32 sets, 256 byte block size, and a 2-way associative will be of size 16 KB.

- Block or Cache-line Size: Block size is the unit of data transfer between main memory and cache. Large cache line can transfer more data whenever a cache miss occurs. It can yield a better hit rate [65].

- Placement Policy: Placement policy determines the cache location in which an incoming block is stored. Flexible policy can result in more hardware cost in addition to increase or decrease in performance due to complex circuitry.

- Replacement Policy: This policy determines which existing cache block will be replaced to create a space for the new block. The following policies - least recently used (LRU), first in first out (FIFO) or random - can be used as a replacement policy. Most of the architectures use LRU.

- Associativity: Cache associativity determines the number of locations that may contain a given memory address. High associativity allows each address to be stored in multiple locations in the cache. This lowers cache misses caused due to conflicts between lines that ought to be stored at same set of locations.

Low associative cache embeds a restriction to the number of locations at which an address can be placed. This may lead to enhance cache misses, but it results in simplified hardware, reducing the amount of space taken up by the cache [17].

- Mapping Function: The mapping function is the algorithm that is responsible for mapping main memory blocks to cache lines. There exist three different types of mapping. They are: direct, associative and set associative.
  - Direct Mapping: This scheme maps each block of main memory to a single cache line.
  - Associative: This scheme is more flexible and can load a main memory block into any line of cache.
  - Set Associative Cache: In In this type of cache there exists number of sets. Each set contains a number of cache lines. A given block maps to any cache line in a given set [86]

3.3 Impact of Cache Size

Current microprocessors are multicore systems, and the number of cores that are integrated into a single processor is likely to increase in the future. One of the challenge faced by multicore processors is to provide adequate memory access for the processor cores. Caches can reduce the requirement of memory bandwidth. But as the number of cores increases, processor designers must find a way to provide required memory bandwidth, without increasing memory access latency. To improve the performance of memory interface and supply data to all cores, the newer processor design integrates the on-chip memory controller with the processor [40]. Comparing with the previous design that required an off-chip memory controller, this solution offers memory access with increased bandwidth and reduced latency [15]. One of the most important factors that affect the execution time of a program is the cache access time [47, 52]. Cache memory is responsible for faster supply of data during execution forming a link between the processing unit and the memory unit. Researchers have shown considerable interest in understanding the impact of cache performance on execution time in order to obtain a better performance of multicore processors [48, 73]. The current advancement in cache memory subsystem in multicore includes addition of more levels of cache as well as the enhancement cache size.

3.4 A New Cache Configuration for Multicore Processors

Benchmarks used today to evaluate the performance of alternative designs are more accurate in estimating the number of clock cycles, cache, main memory and I/O bandwidth than they were before. SPLASH2 is one such benchmark used by the researchers to estimate the performance. It has concentrated workloads based on real life applications,and is a descendant of the SPLASH benchmark.

The simulator used is Multi2Sim [78] which includes features to support separate functional and timing simulation, simultaneous multithreading processor (SMT), multiprocessor support and cache coherence. Multi2Sim is an application-only tool intended to simulate x86 binary executable files. The performance of different multicore systems are analyzed for different processor organization by varying the L1 and L2 cache size. In our study the number of cores in the architecture is kept constant at eight and the size of L1 and L2 cache have been varied. We have performed the simulation by varying the L2 cache size for six different L1 cache size ranging from 16 KB to 512 KB. The performance of these configurations is evaluated with Fast Fourier Transform (FFT), Cholesky and Barnes from the SPLASH2 benchmark suite [7]. In the proposed work, we varied the size of L1 cache keeping the block size fixed at 256 byte, and associative to two-way. For a given cache size teh number of sets is varied from 32 to 1024.

The proposed memory configuration description is given in Table: 3.1, for L1 cache size 512 KB and L2 cache size 8 MB.

3.5 Simulation Results

Simulation is performed using Multi2Sim simulator. Parameters considered for simulation is shown in Table 3.1. In the simulation for a fixed L1 cache size, the size of L2

Table 3.1: Proposed Cache Configuration of Multicore Processor

Processor Frequency = 1000 MHz Default Output Buffer Size = 1024 packets	Input Buffer Size = 1024 packets Defaut Bandwidth = 1024 bytes per cycle

CacheGeometry geo-d-l1
Sets = 1024
Assoc = 2
BlockSize = 256 bytes
Latency = 2
Policy = LRU
Ports = 2
	CacheGeometry geo-i-l1
	Sets = 1024
	Assoc = 2
	BlockSize = 256 bytes
	Latency = 2
	Policy = LRU
	Ports = 2

CacheGeometry geo-l2
Sets = 8192
Assoc = 4
BlockSize = 256 bytes
Latency = 20
Policy = LRU
Ports = 4
	Module mod-mm-2
	Type = MainMemory
	BlockSize = 256 bits
	Latency = 200
	HighNetwork = net0
	HighNetworkNode = n4

cache is varied between 1MB to 8 MB. The above simulation is repeated for different size of L1 cache. The interconnection network considered in the simulator is MPIN, shown in Figure 4.2.

The execution time of FFT, Barnes and Cholesky versus L2 cache size is shown in Figure 3.2, 3.3, and 3.4 respectively. It is observed from Figure 3.2 that the execution time of FFT decreases with increasing the size of L2 cache. It is also observed that for a given L1 cache size, the execution time decreases with increasing the size of L2 cache. The decrease is significant, when the L1 cache size is increased from 128 KB to 256 KB. Beyond 256 KB the decrease is marginal. We also observed that beyond 8MB of L2 cache, there is no improvement in the execution time for any fixed size of L1 cache.
We also, observed that the execution time pattern of Barnes and Cholesky is similar with FFT. However, in Barnes and Cholesky, for L1 cache size greater than 64 KB, gain in the execution time is less. The execution time of Cholesky is minimal for L1 cache size of 512 KB. For L2 cache size greater than 4 MB, the execution time of Barnes and Cholesky remain almost constant. From the above figures, we can conclude that 4 MB L2 cache is sufficient for any size of L1 cache.
Execution time obtained for FFT by varying L2 cache size and keeping L1=16 KB, L1 = 32 KB, L1 = 64 KB, L1 = 128 KB, L1 = 256 KB and L1 = 512 KB has been illustrated

by Figure 3.2. Execution time reduces with increase in the L2 cache size. But beyond L2 = 8 MB the execution time displays a constant behavior. Similarly as the L1 cache size increases from 16 KB to 512 KB, the execution time keeps on reducing for each L2 cache size. When the L1 cache size reaches 512 KB, the execution time is the least for every L2 cache size. For L1, = 512 KB and L2 = 8 MB, the execution time is the least in comparison with all other variation of L1 and L2 cache size.

Similar experimental setup was utilized to execute another benchmark program from the SPLASH2 benchmark suite. Figure 3.3 and 3.4 illustrates the execution time variation with different combinations of L1 and L2 cache size ranging from 16 KB to 512 KB for L1 cache, and 512 KB to 8 MB for L2 cache on executing barnes and cholesky benchmark program respectively. For this benchmark too, it has been observed that as the cache size increased the miss rate reduced due to which a better execution time has been obtained. Performance enhancement has not been much beyond the L2 cache size of 8 MB.

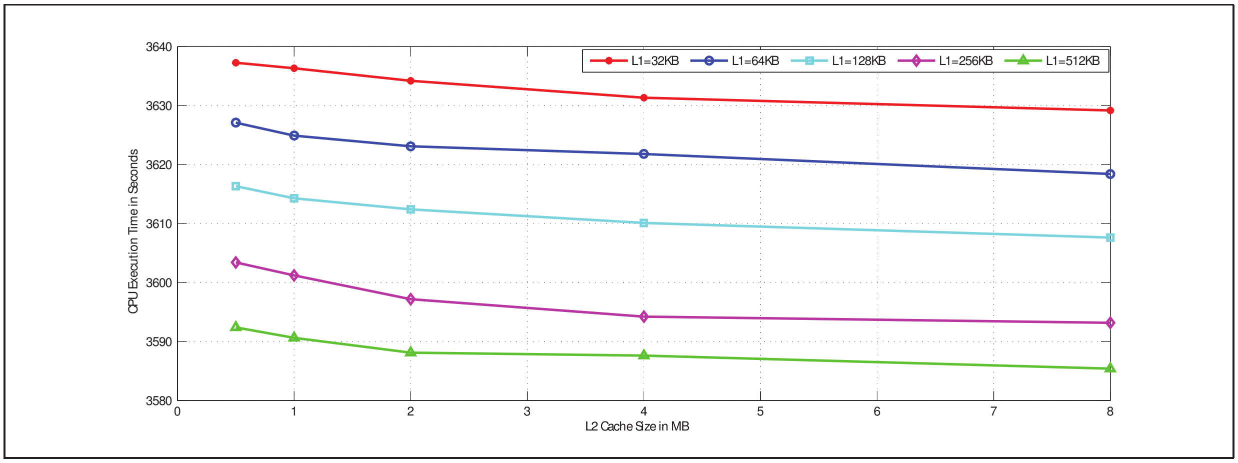

Figure 3.2: Execution time of FFT vs. L2 cache size

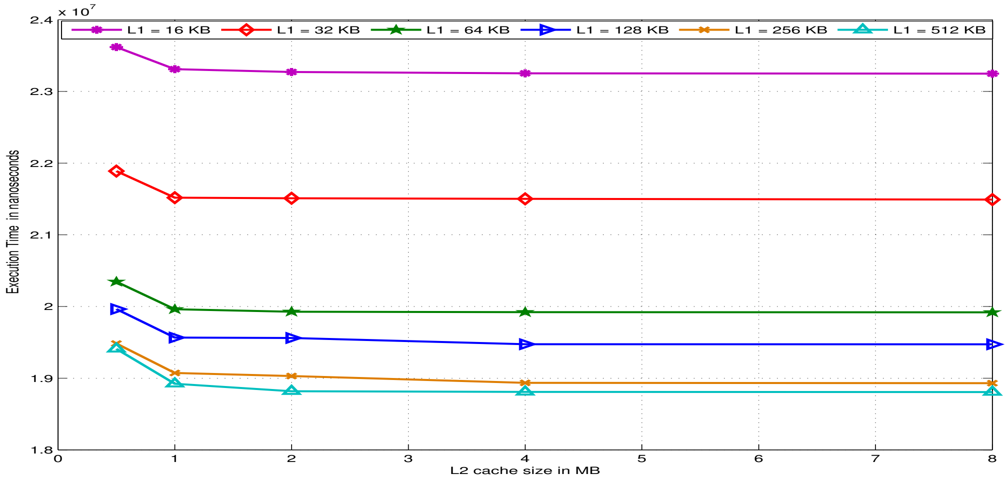

Figure 3.3: Execution time of Barnes vs. L2 cache size

3.6 Summary

In this chapter, a novel cache configuration for performance enhancement of multicore processor has been proposed. The new cache configuration provided a better performance in comparison to the existing cache configurations. The MPIN model with eight cores has been used for this performance analysis, although it can be extended to handle other higher core counts and processor architecture. This cache configuration can be extended to propose a novel cache configuration for multicore mobile processors.

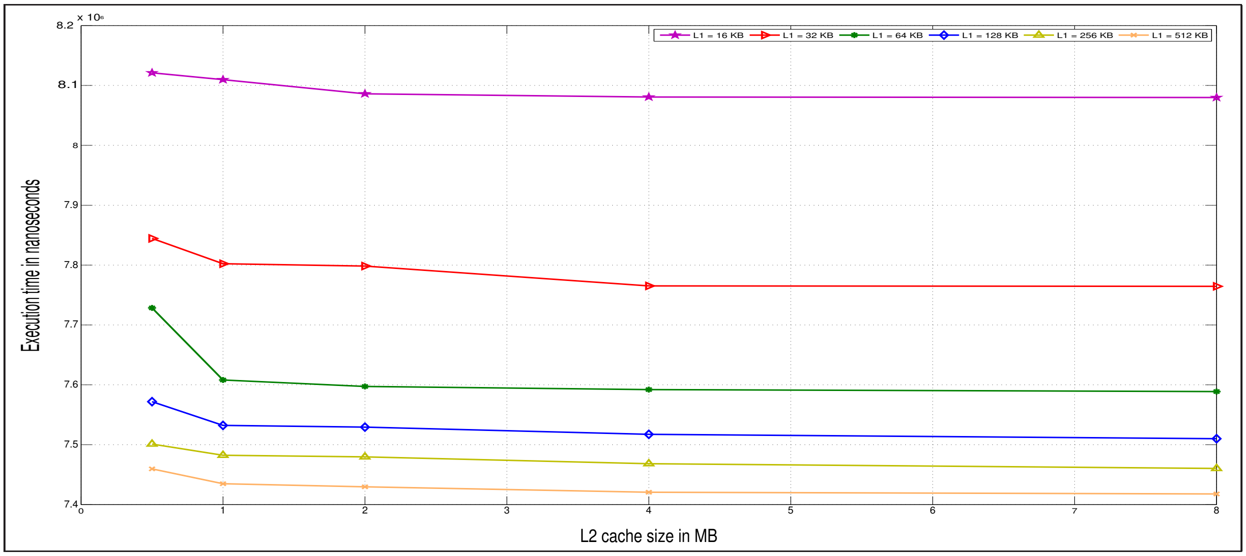

Figure 3.4: Execution time of Cholesky vs. L2 cache size

Chapter 4

An Interconnection Network-on-Chip for Multicore Processor

4.1 Introduction

Interconnection among the cores in a multicore processor has posed a great challenge. With increase in the number of cores, the traditional on-chip interconnect like bus and crossbar has proved to be less efficient. Moreover, they also suffer from poor scalability. To overcome the scalability and efficiency problem in these traditional interconnects, ring based design has been proposed. However, with steady growth in the number of cores, the ring based interconnect has also become infeasible. This necessitates, the designer to look for a novel way of interconnect among the cores without degrading the efficiency and scalability. In this chapter, a novel interconnect network for multicore processor is proposed. It is compared with multicore processor with internal network (MPIN). Multi2Sim simulator and SPLASH2 benchmark suite is used for performance evaluation. It was observed that the proposed interconnect is more efficient than the existing MPIN. It is also scalable too.

In processor design the constraints with respect to power consumption, clock frequency and heat dissipation have made the chip designers to evolve from improvements in single-core processor to integration of multiple cores on a single chip. The trend that has been used for enhancement of performance is to enhance the number of cores per chip [87]. This enhancement has led to the concept of network-on-chip (NoC). Before this concept, system-on-chip (SoC) took the aid of complex traditional interconnects like bus structures for connection between the cores to memory and I/O.

Traditional bus structures were improved, to be used as interconnect in the multicore processors. However, with enhancement in the number of cores, use of bus as interconnect proved inefficient with increasing complexity. Moreover, bus does not scale well with increase in the number of cores. To address the scalability problem in multicore, NoC is used [12, 42].

In this chapter, we analyze the effect of interconnect on the multicore processors and have proposed a novel scalable on-chip interconnection mechanism for multicore processors.

4.2 Architecture and Background

Various work in current literature has explored the multicore architecture utilizing various performance metrics and application domain. Researchers like Bucker et al. [12] in their work have analyzed the effect of NoC on power consumption of multicore processors. They have discussed the challenges as well as progress in research and development of NoC for multicore processors. Zhou et al. [88] have proposed a new scheme for cost minimization named *Interconnect Communication Cost Minimization Model (ICCM)*. They have demonstrated an average of 50 % reduction in the communication cost, average of 23.1 % throughput improvement, and 35 % dynamic power reduction with the aid of their optimization mechanism.

Tullsen et al. [54] have analyzed a single-ISA heterogeneous multicore architecture for multi-threaded workload performance. The objective was to analyze the performance of multicore architectures for multi-threaded workloads. This section details the benefits of variation in the interconnection network in the multicore architecture with multi-threaded workloads.

Through various works, performance has been analyzed in both single core and multi-core architectures. Julian et al. [13] determined the relationship between performance and memory system in single core as well as multicore architecture. They have evaluated multiple performance parameters like cache size, core complexity. The authors have also discussed the effect of variation in cache size and core complexity across the single core and multicore architecture.

Multicore architecture with multiple types of on-chip interconnect network [24, 80] are the recent trend of multicore architecture. This type of architecture have different types of interconnect network with different core complexity and cache configuration. A few of the architecture of this type has been described in the subsequent sections.

4.2.1 Metrics of Interconnection performance

The performance of interconnection networks can be described using a few metrics. These metrics have been cited below:

- Connectivity, or degree: This is the number of nodes that can be reached from a particular node in a single hop.
- Diameter: This gives the maximum delay that can occur during the passage of a message from one processor to another.
- Distance: The distance between two nodes is given by the total number of hops in the shortest path between two nodes.
- Average Distance: It is given by the the equation

  $$d_{avg} = \frac{\sum_{d=1}^{r} (d - N_d)}{N - 1}$$

  where N: The number of nodes, N_d is the number of nodes at distance d apart and r is the diameter.
- Bisection Bandwidth: It is the minimum number of wires that need to be disconnected to divide the network into two equal halves.

4.2.2 Switches

Initially to establish communication in the multicore processor, buses were used as a shared fabric. In a bus, the communication is always in broadcast, in which, a message going from entity A to entity B, is seen by all other entities. In addition to it multiple messages cannot be in transit simultaneously. But, when the cores and memory are placed on two either sides of the die, then it becomes a compelling choice to take the aid of crossbar interconnect for better communication. Even though crossbar are more costlier than buses with respect to the wiring overhead, still they enable simultaneous transmission of multiple messages, which in turn results in enhanced network bandwidth.

A crossbar circuit has N inputs and establishes a connection between these N inputs and M possible outputs. The circuit is setup as a grid of wires. At every point where the wires intersect, a pass transistor serves as a crosspoint connector which can short the two wires when enabled, which in turn connects the input to the output. We have small buffers at the crosspoint which can store the messages for a temporary period of time at times of contention for the target output port. A centralized arbiter is responsible to control the crossbar. This arbiter takes requests for output port from the incoming messages and computes a viable assignment of input port to the output port connections. The network model as shown in the proposed architecture in Figure 4.3 consists of set of nodes and switch nodes. Few links that connects the input and output buffers of a pair of nodes is also included in this network. The nodes existing in this network is classified as the end nodes and switch nodes. An end node is able to send or receive packets to or from another end node. Switches are used only to forward the packets between the switches or end nodes [44]. A link is particularly used for connection between the end nodes and switches. The internal architecture of a switch is displayed in the Figure 4.1. This switch includes a crossbar which connects each input buffer with every other output buffer. A packet at any input buffer can be routed to any of the output buffer at the tail of the switch [79].

4.2.3 Multicore Processor with Internal Network

The architecture of a multicore processor with internal network is shown in the Figure 4.2. This architecture is primarily targeted for the commercial server class applications that have a low degree of instruction level parallelism (ILP) but a high degree of thread level parallelism (TLP). It has a private L1 cache dedicated for each core. L1 caches

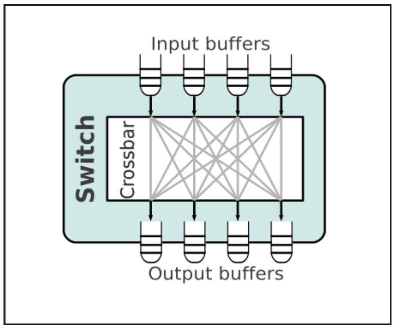

Figure 4.1: Switch Architecture Model

are connected to two L2 cache banks through a switch [78]. Each core is capable of executing eight threads independently. At every cycle, the thread selection stage of pipeline chooses one of the eight threads for execution. An instruction from the selected thread is fetched and pushed onto the beginning of the pipeline. The thread selection policy aims at switching threads in every cycle giving a fine-grained interleaving of threads in the pipeline [14].

Interconnect

MPIN architecture uses a crossbar switch as interconnection network. A core can communicate with another core or L2 cache only through the switch. The crossbar allows up to eight simultaneous accesses of the L2 cache. Four memory controllers are used to interface L2 cache with main memory.

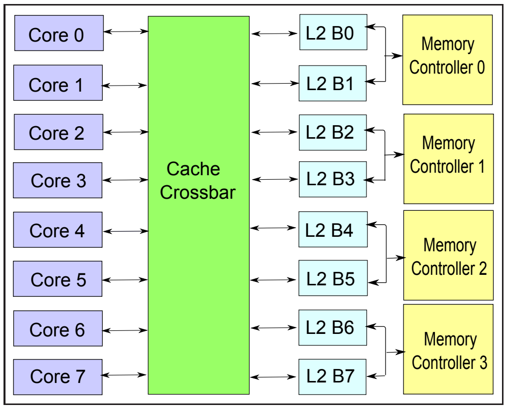

Figure 4.2: Multicore Processor with Internal Network

Demerits

The crossbar switch becomes a bottleneck, with increase in the number of cores. Two L2 cache shares the same main memory controller. Therefore, a contention to access the main memory takes place, if there is misses in both the L2 cache. Moreover, the on-chip memory controller imposes restriction, that all DDR memory positions must contain equal number of DIMMs of same capacity, speed and from the same manufacturers [4, 44].

4.3 A novel Interconnection Network-on-chip for Multicore Processor

The proposed interconnect architecture for multicore processor is explained in this section. This architecture is named as Interconnection Network-on-Chip (INoC). In the proposed architecture, each core has its own private L1 cache. The L2 cache is sharable between four cores through a switch. Cores sharing the same L2 cache, communicate through their sharable L2 cache. Cores with different L2 cache which communicate through their respective L2 cache and an intelligent network, which we call Interconnection network-on-chip (INoC). INoC has a set of buffers and memory controller. Each L2 cache has it own private buffer in the INoC. Similarly, each memory module has its own private buffer MB. Data read from the memory module are stored in the

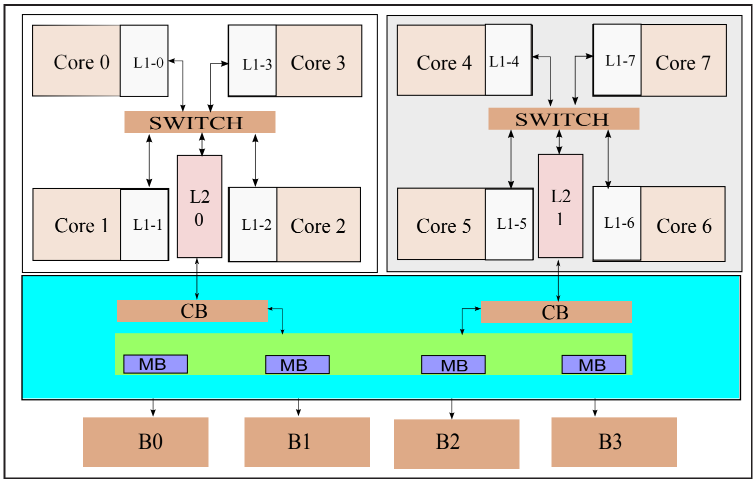

Figure 4.3: Multicore Processor with Interconnection Network-on-Chip

MB; before it is written into the corresponding L2 cache. The requests for which there were L2 cache miss, are stored in CB. In case of write miss, it contain the corresponding cache block and its address. For a read miss it contain the address from which the block is to be read from the memory. Having two independent buffers allows the L2 cache and each memory module to operate simultaneously. The scalability of the interconnect is enhanced without major power or energy alterations because of the usage of four lower configurations of the crossbar resulting in lower power consumption. Multiple lower configured crossbar proves to be economical as well as more scalable as compared to high configurations of crossbar [20].

Memory controller determines which memory module is to be read/written. Since there are more than one memory module the memory controller can perform more than one operation at the same time. For example, suppose there are two L2 cache miss from two different L2 cache. If the two misses, addresses to different memory module then the memory controller schedule two operation simultaneously. This increases the memory throughput. If both the misses are addressed to the same memory module, then the memory controller serialize the memory access. The operation performed by the L2 cache is stored on the buffer CB, and that performed by memory is stored in MB. Since, the operation performed by the cache and memory involved different buffers, the average memory access time is reduced to a certain extent.
Figure 4.3 shows the diagram for eight core. The number of cores in the proposed architecture is to be multiple of four. So that four cores can be connected through a switch to a L2 cache.

4.3.1 Architectural Communication

In this architecture coherence is maintained through directory-based MOESI protocol [75]. Each block in L2 cache contains fields like tag, data, and status. A block also has a directory entry which contains two fields. One of the field represents the state in which this block is with a L1 cache. The state can be exclusive, owned or modified. While the other holds the bitmap with as many bits as L1 cache. The bits in the bitmap are set to one corresponding to the caches having a copy of this block.

The MOESI protocol enforces coherence with the concept of write exclusiveness; before writing into a cache block, the protocol first ensures that the target cache owns an exclusive copy of the block.

4.3.2 Cache Directories

To enforce coherence between a cache in level 2 and a cache closer to the processor in level 1, a directory is attached to the level-2 cache. Each core has private level 1 data cache and level 1 instruction cache. This split cache offers two memory ports per clock cycle thereby avoiding the structural hazard and having a better average memory access time than the single unified cache discussed in [43]. A directory has one entry for each block present in its corresponding cache. Together, a cache block and its associated directory entry contain the following fields:

- State: The state of the block can be any of the five MOESI states (Modified, Owned, Exclusive, Shared, or Invalid). To encode these five possible states, cache required a three bit field.
- Tag: If the block is in any state other than I, this field uniquely identifies the address of the block in the entire memory hierarchy. In a system with 32-bit physical addresses and a set-associative cache with S number of sets and block size of B bytes, the number of bits needed for the tag can be given by the expression below.

$$Number of Tag bits required = 32 - \log(B) - log(S) \tag{4.1}$$

- Data. Block of B bytes.
- Owner. This field contains the identifier of the higher-level cache that owns an exclusive copy of this block, if any. A special value is reserved to indicate that no exclusive copy of the block exists. For a L2 cache with n L1 caches connected to it, this field has $\log_2(n+1)$, bits.

- Sharers. Bit mask representing the higher-level caches sharing a copy of the block, either exclusive or non-exclusively. This field has *b* bits for an L2 cache with *b* number of L1 caches connected to it. The first three fields contained in cache blocks, while the three last fields indicate a directory entry. L1 caches do not attach a directory, since they lack higher-level caches to keep coherence among. Thus, the presence of a block in a first-level cache only requires storing fields State, Tag, and Data.

Let us assume block A holds the sharers list as core-0, core-2. In such a case if core-0 sends a write request for block A, on receiving this request from core-0 for block A, the directory detects that the block is valid but core-0 does not have the permission to write to this block. Thus, a write request is send to the memory module containing block A and core-0 is stalled. Now the directory issues invalidate requests to the other cache holding this block which is core-2 cache in this case. Now the cache in core-2 on receiving this invalidate request, sets the appropriate bits to indicate that the block A is invalid and an acknowledgment is sent back to the directory. On receiving this acknowledgment the directory clears the sharers list of core-2 entry and sends the write permission to core-0. Now, core-0 receives the write permission message and the sharers list is updated and core-0 is reactivated.

Table: 4.1 shows the configuration to create mulicore processor with INoC in Multi2Sim. In this configuration cache geometry geo-d-l1, geo-i-l1 represents L1 data cache and L1 instruction cache configuration respectively for each core. Cache geometry geo-l2 represents L2 cache configuration for the entire architecture. Module mod-mm-o through mod-mm-3 represents the four different main memory banks for the considered architecture. Network net-l1-l2-0 and net-l1-l2-1 represents the network configuration between L1 cache with L2-0 and L2-1 of multicore processor with INoC as shown in Figure 4.3.

Size of L1 and L2 cache is varied by keeping the block size and associativity constant, and increasing the number of sets in the cache. In our simulations the network configuration as well as the main memory configuration remains unaltered. This lets us determine the effect cache size holds on performance of multicore processors.

Table 4.1: Code Level Configuration in Multi2Sim to create multicore processor with INoC interconnect

Processor Frequency = 1000 MHz	Input Buffer Size = 1024 packets
Default Output Buffer Size = 1024 packets	Defaut Bandwidth = 1024 bytes per cycle
CacheGeometry geo-d-l1	Module mod-mm-0
Sets = 32	Type = MainMemory
Assoc = 2	BlockSize = 256 bytes
BlockSize = 256 bytes	Latency = 200
Latency = 2	HighNetwork = net0
Policy = LRU	HighNetworkNode = n2
Ports = 2	AddressRange = ADDR DIV 256 MOD 4 EQ 0
CacheGeometry geo-i-l1	Module mod-mm-1
Sets = 32	Type = MainMemory
Assoc = 2	BlockSize = 256 bits
BlockSize = 256 bytes	Latency = 200
Latency = 2	HighNetwork = net0
Policy = LRU	HighNetworkNode = n3
Ports = 2	AddressRange = ADDR DIV 256 MOD 4 EQ 1
CacheGeometry geo-l2	Module mod-mm-2
Sets = 8192	Type = MainMemory
Assoc = 4	BlockSize = 256 bytes
BlockSize = 256 bytes	Latency = 200
Latency = 20	HighNetwork = net0
Policy = LRU	HighNetworkNode = n4
Ports = 4	AddressRange = ADDR DIV 256 MOD 4 EQ 2
Network net-l1-l2-0	Module mod-mm-3
DefaultInputBufferSize = 1024	Type = MainMemory
DefaultOutputBufferSize = 1024	BlockSize = 256 bytes
DefaultBandwidth = 256 bytes per cycle	Latency = 200
Network net-l1-l2-1	HighNetwork = net0
DefaultInputBufferSize = 1024	HighNetworkNode = n5
DefaultOutputBufferSize = 1024 packets	AddressRange = ADDR DIV 256 MOD 4 EQ 3
DefaultBandwidth = 256 bytes per cycle	

4.3.3 Impact of Cache Size

To study the impact of cache size on multicore processor, the number of cores are kept constant at 32 and the size of L1 and L2 cache was varied. L2 cache size is varied first keeping the L1 cache size constant. Then L1 cache size was varied keeping L2 cache size constant. The execution time for FFT, Cholesky, and Barnes benchmark program of the SPLASH2 benchmark suite was analyzed. Figure 4.4 shows the execution time of FFT benchmark with variation in L2 cache size for different size of L1 cache in the proposed architecture. Figure 4.5 shows the variation in execution time of Cholesky benchmark with variation in L2 cache size for different size of L1 cache in the proposed architecture.

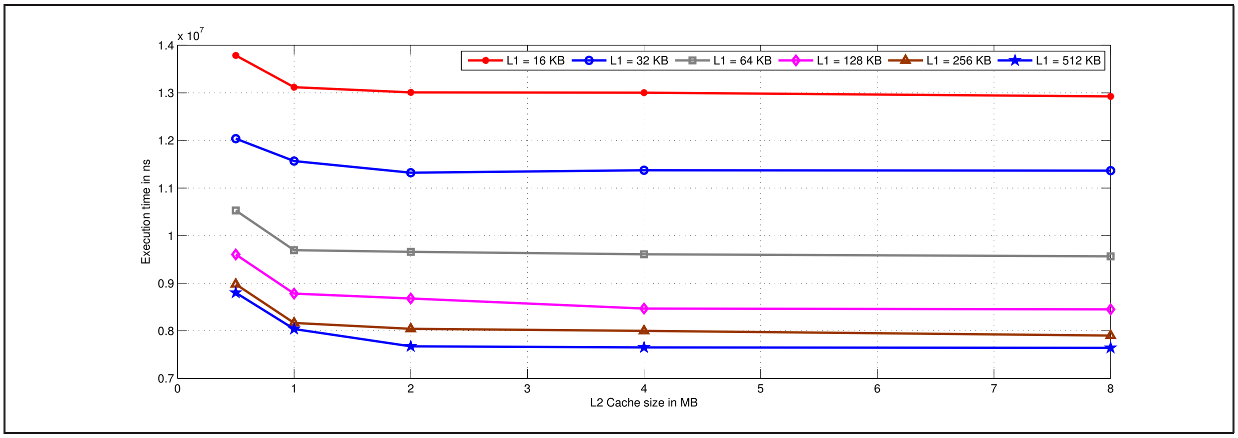

Figure 4.4: Execution time of FFT vs. L2 cache size for different L1 cache size

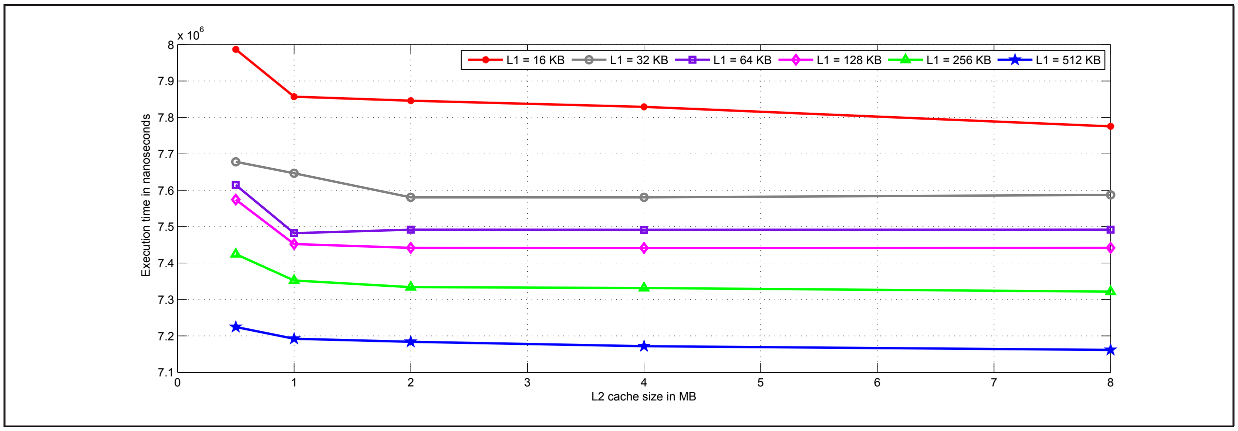

Figure 4.5: Execution time of Cholesky vs. L2 cache size for different L1 cache size

Figure 4.6 shows the variation in execution time of Barnes benchmark with variation in L2 cache size for different size of L1 cache in the proposed architecture. In the above simulation, the proposed INoC is used. It is observed from the above Figures that the execution time of considered benchmark program decrease with increase in the size of both L1 and L2 cache. This is because with increase in the size of caches, the chances of each hit increases.

We observed that beyond eight MB size for L2 cache the execution time remains almost constant. From Figures 4.4, 4.5 and 4.6 we conclude that best performance is achieved when L1 cache is 512 KB and L2 cache is 8 MB, for proposed architecture with eight cores.

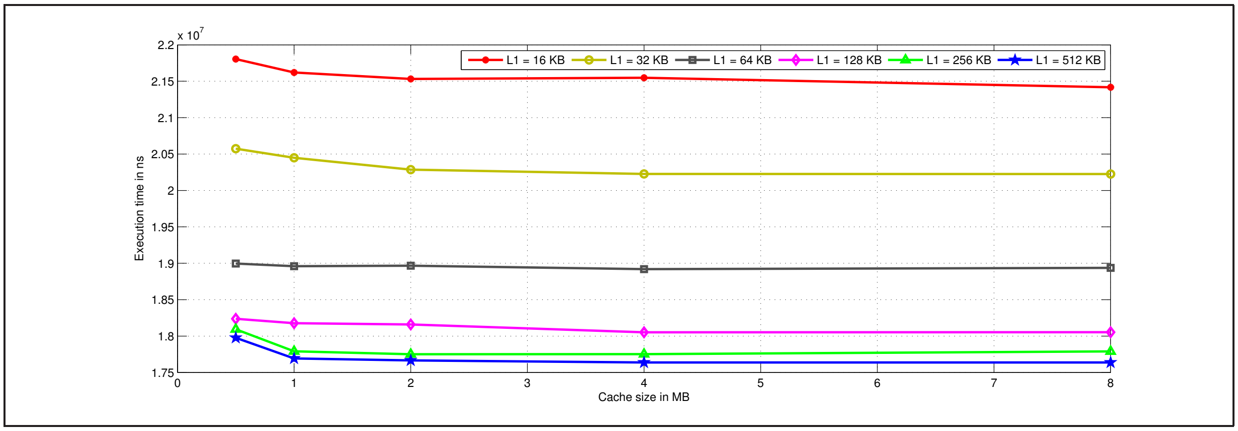

Figure 4.6: Execution time of Barnes vs. L2 cache size for different L1 cache size

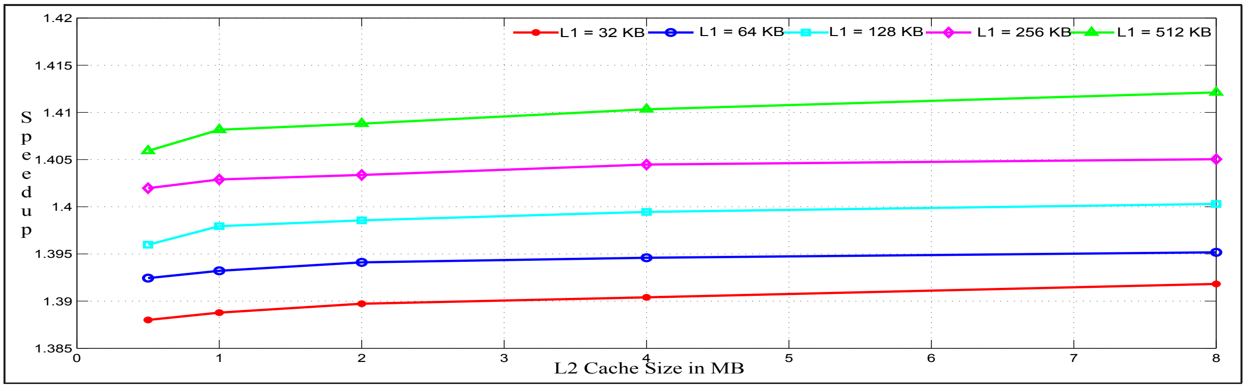

Figure 4.7: Speedup vs. L2 cache for the proposed interconnect

The speedup for multicore architecture with proposed Interconnect has been given in the Figure 4.7. Expression for speedup is discussed in Section 2.4 of Chapter 2. The old execution time obtained for the benchmark program for a single core pentium 4 processor with 16 KB L1 cache and 2 MB of L2 cache.

4.3.4 Impact of Number of Cores

The chances of conflict increases as the number of cores increase. Figure 4.8 gives the scalability of INoC interconnect. Execution time is lowered as we move from 2 to 128 cores with our proposed interconnect. Even though INoC exhibits slightly better, performance than MPIN up to 64 core, it is expected to outperform MPIN with more than 64 cores. Scalability of our proposed interconnection is illustrated by Figure 4.8. It is observed that execution time decrease with increase in the number of cores. Thus we conclude that the proposed interconnect is scalable. In Figure 4.9 the execution

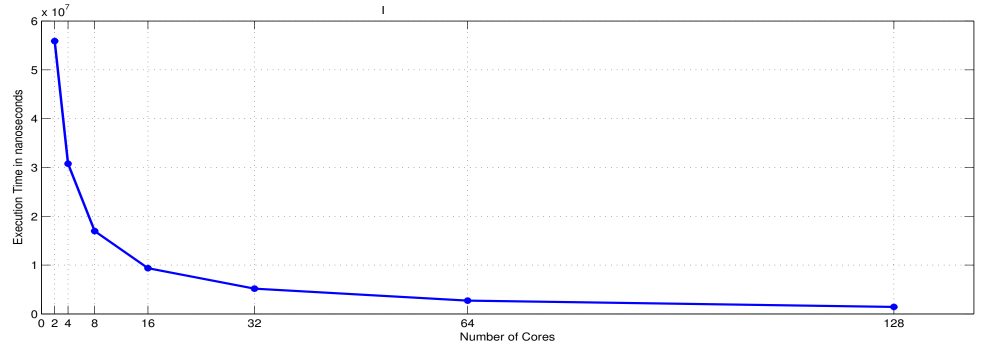

Figure 4.8: Execution time vs. number of cores showing scalability of INoC

time of FFT benchmark program over MPIN, and INoC using the same simulation framework is depicted graphically. Here, the L1 cache size is kept constant at 512 KB, and L2 cache size at 8 MB. The number of cores is varied from 2 to 128. We have only shown the performance variations for 16, 32, 64 and 128 core. It was observed that, there is gradually decrease in the execution time of FFT with increasing number of cores.

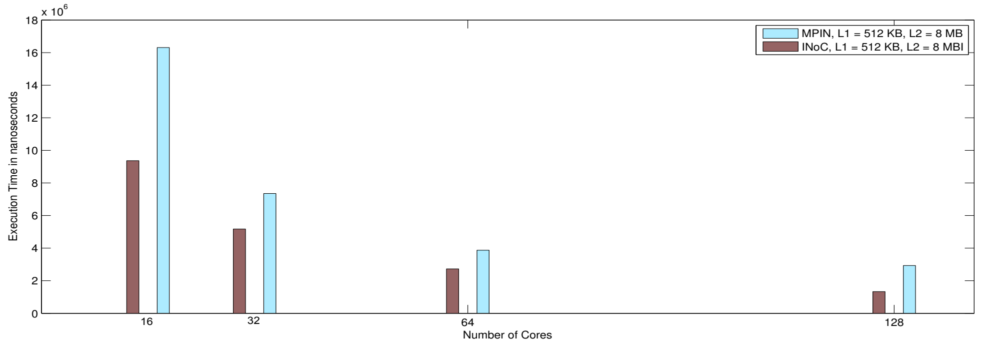

Figure 4.9: Execution Time of FFT vs. Number of cores in MPIN and INoC interconnection network, L1 cache size 512 KB and L2 cache size 8 MB

4.4 Results and Discussion

The performance of INoC is compared with the existing MPIN architectures described in previous section. Figure 4.10 illustrates performance difference between MPIN and INoC for executing FFT benchmark with 16 KB L1 cache and L2 cache of size varied from 512 KB to 8 MB.

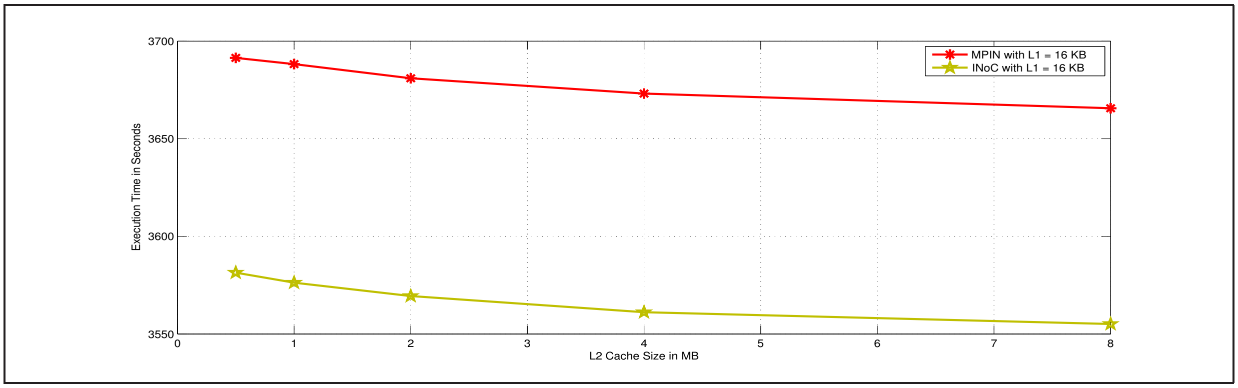

Figure 4.10: Execution Time of FFT vs. L2 cache in MPIN and INoC interconnect with 16 KB L1 Cache

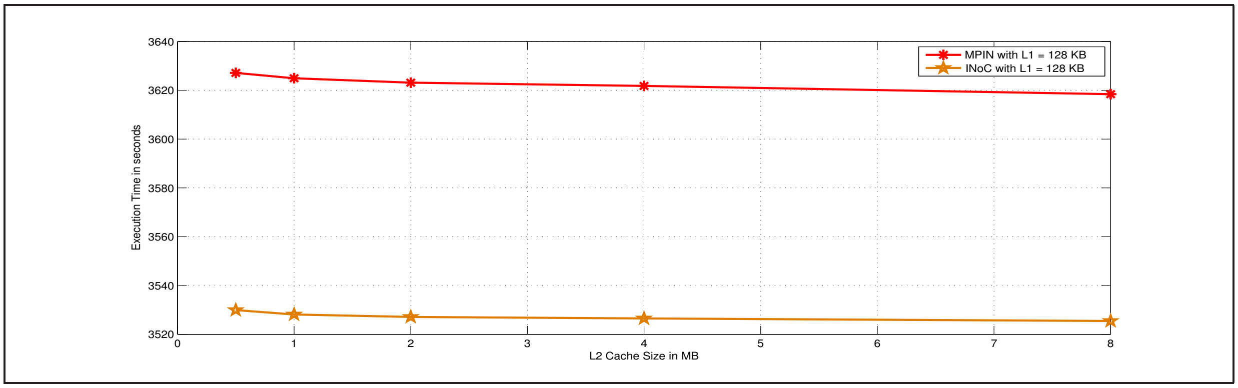

Figure 4.11: Execution Time of FFT vs. L2 cache in MPIN and INoC interconnect with 128 KB L1 cache

Similarly, Figures 4.11 and 4.12 illustrates the performance difference on executing FFT benchmark with L1 cache size of 128 KB and 512 KB respectively. The execution time of FFT in the proposed interconnect is lesser as compared with MPIN. For the considered architecture with INoC interconnect, best performance is achieved when the L1 cache size is 512 KB and L2 cache size is 8 MB.

4.5 Summary

In this chapter, we have proposed a novel interconnect for multicore processors. We have named our interconnect as *Interconnection Network on Chip* (INoC). The proposed interconnect is compared with MPIN. For comparison, we consider the same processor architecture i.e. equal number of cores, equal sizes of L1 and L2 caches, but with different interconnection network between processor and memory. Three benchmark

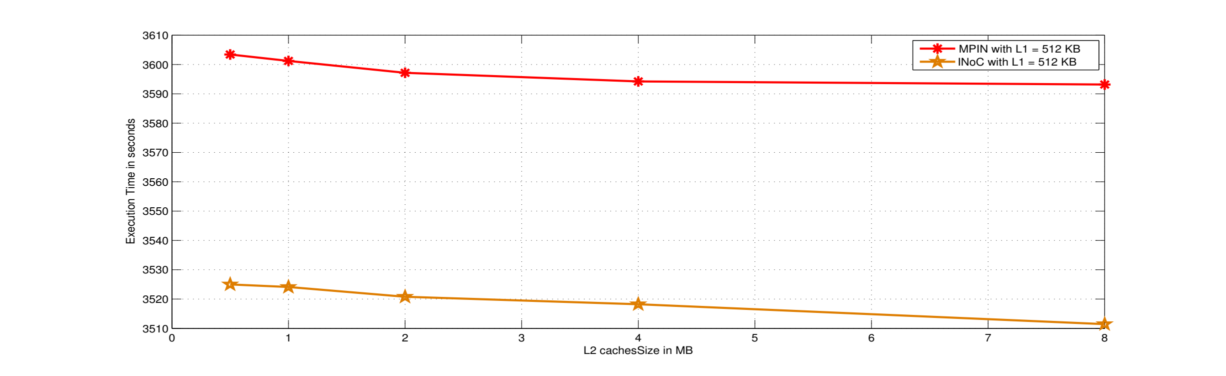

Figure 4.12: Execution Time of FFT vs. L2 cache in MPIN and INoC interconnect with 512 KB L1 cache

program FFT, Barnes, and Cholesky is executed to compare the performance of MPIN and INoC. From the execution time of benchmark program, we conclude that the proposed INoC interconnect yield better performance.

Chapter 5

Cache Configuration for Multicore Mobile Processor

In this part of our dissertation, with varying cache features in the mobile processor we have addressed its performance evaluation and enhancement. In order to achieve enhanced cache performance and lower chip area, we focused on the use of a novel cache configuration in mobile multicore processors.

5.1 Background

In computing society a technical break through was provided by multicore processors. Although in general-purpose processors area the processors with multiple cores have already flourished, yet the mobile processor companies have not accelerated their processors with multiple cores except for a few like Intel Atom [5, 56], ARM Cortex [70]. Certain issues related to its design are countered not only in computer system processors but also critically more in mobile processors. More distinctively in battery-operated mobile devices, heating as well as power problems are major issues to be dealt with. In order to match the mobile technologies with mobile processors, issues needs to be vigorously investigated and specific solutions has to be generated for design of better mobile processor in future. At present, in the domain of multicore mobile processor dual research directions are pragmatic, one being a group on embedded cellular-technology based and the other one is x86-based. Increasing the cache size to its double generally is the easiest way to escalate the performance, but will affect the chip area in addition to power consumption. With time trend of multicore processors are dominating and the cache structures are getting more and more complicated and sophisticated, thus larger L2 cache is required which can be shared across. Appreciating the integrating technologies, multicore mobile processors integrated onto a single chip though tiny in size can integrate bigger caches, yet architectural issues related to its design and fabrication cannot be ignored. Issues that are critical to the design of mobile processors are power and heat dissipation. The above issues gets worsened when more cores are put into a mobile processor chip with a smaller form factor and without cooling fan. Cache memory also consumes, a substantial amount of power. Re-organization and reanalysis of cache configuration of mobile processors is required to achieve better cache performance and lower chip area in addition to reduction power consumption. Cache size as well as set-associativity are two of the features on which the cache performance is dependent. Doubling the cache size, can increase the cache performance considerably. But at the cost of high hardware, more power and larger chip area. Moreover, increasing the cache size may affect the size of the mobile device and its battery running time. Set-associativity affects the cache miss rate, higher set-associativity can achieve a lower cache miss rate [45].

5.2 Shared Cache Features

The performance of shared L2 cache has become more critical with wide spread usage of mobile internet devices (MID) and the merger of multicore concepts with tiny mobile processors [45]. The size of L2 cache in mobile devices typically ranges from 64 KB to 8 MB [35, 41]. For ARM Cortex-A8 it may range up to 2 MB; while for ARM Cortex-A9, it may range up to 8 MB [70]. Considering multicore mobile processors, this research aims in improvement of cache performance with lower size of shared L2 cache. Size of the L1 cache is of 32 KB for most of the mobile processors [35, 41, 70].

Doubling the cache size has been a normal practice to improve the cache performance To reduce the cache miss rate different replacement policies have been used in multicore processors [26, 35]. However, the effectiveness of these alterations differs for different applications.

5.3 A New Cache Configuration for Multicore Mobile Processor

An increase in set-associativity reduces cache miss rate. However, the improvement in cache miss rate is lower in comparison to the improvement when cache size is increased. The miss rate can be reduced without altering the cache size too. In the proposed configuration the set-associativity is altered while keeping the block size constant and varying the number of sets. We intend to define a cache configuration which will yield better performance with lowered size cache. Power consumption can be lowered, if the same performance can be achieved with a cache of lower size. For mobile processors which require small-sized cache the proposed cache configuration is shown in Table 5.1. The above configuration is obtained through extensive simulation.

Table 5.1: Proposed Cache Configuration of Multicore Mobile Processor

CacheGeometry geo-d-l1
Sets = 32
Assoc = 2
BlockSize = 256 bytes
Latency = 2
Policy = LRU
Ports = 2
	CacheGeometry geo-i-l1
	Sets = 32
	Assoc = 2
	BlockSize = 256 bytes
	Latency = 2
	Policy = LRU
	Ports = 2

CacheGeometry geo-l2
Sets = 128
Assoc = 16
BlockSize = 256 bytes
Latency = 20
Policy = LRU
Ports = 4
	Module mod-mm-2
	Type = MainMemory
	BlockSize = 256 bytes
	Latency = 200
	HighNetwork = net0
	HighNetworkNode = n4

5.4 Simulation and Results

For simulation we have considered a two core mobile processor. The size of L1 cache is kept 16 KB. The associativity and size for L2 cache is varied to study the performance. Initially associativity of L2 cache is varied from 2-way, to 16-way keeping the block size constant. The size of L2 cache is taken to be 64 KB and the number of sets is varied from 16 to 256. Next, we performed the simulation for different size of L2 cache. To obtain a 128 KB L2 cache, the number of sets was varied from 32, 64, 128, 256, 512 for

16-way, 8-way, 4-way, and 2-way set associativity respectively. Similarly, the number of sets was varied upto a maximum value of 2048 to obtain a 2-way set-associative 1 MB L2 cache. The minimum value for number of sets was 16 for a 64 KB 16-way set-associative L2 cache. In each case FFT benchmark of the SPLASH2 suite was executed. The execution time was taken into consideration to analyze the performance of multicore mobile processor.

Figure 5.1 shows the variation in execution time with different set associativity for a 64 KB L2 cache on executing FFT benchmark. As the set associativity increases the execution time reduces. A 64 KB 16-way set-associative L2 cache gives the best performance in executing FFT on a 2-core processor with 16 KB L1 ccache.

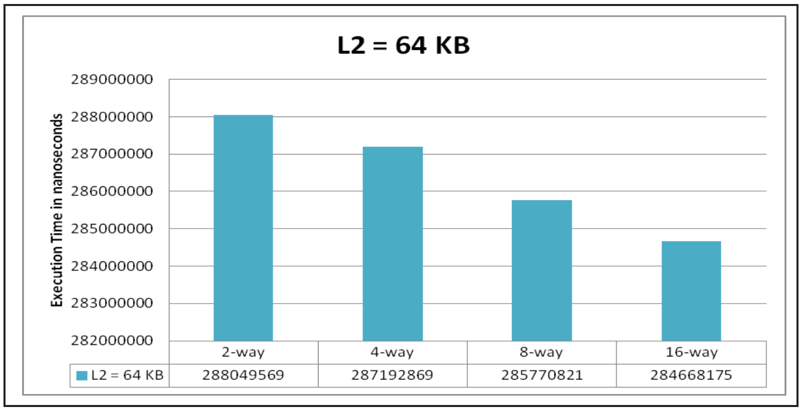

Figure 5.1: Execution time vs. set associativity for multicore mobile processor with 64 KB L2 Cache

Figure 5.2, shows the execution time of FFT in a 2-way, 4-way, 8-way and 16-way, set-associative L2 cache of size 128 KB. For a multicore mobile processor with 2 cores, 16 KB L1 cache, and 128 KB L2 cache, the best performance can be observed for a 16-way set associative L2 cache.

Figure 5.3, shows the execution time of FFT in a 2-way, 4-way, 8-way and 16-way set associative L2 cache of 256 KB. For a multicore mobile processor with 2 cores, 16 KB L1 cache, and 256 KB L2 cache, the best performance can be observed for a 16-way

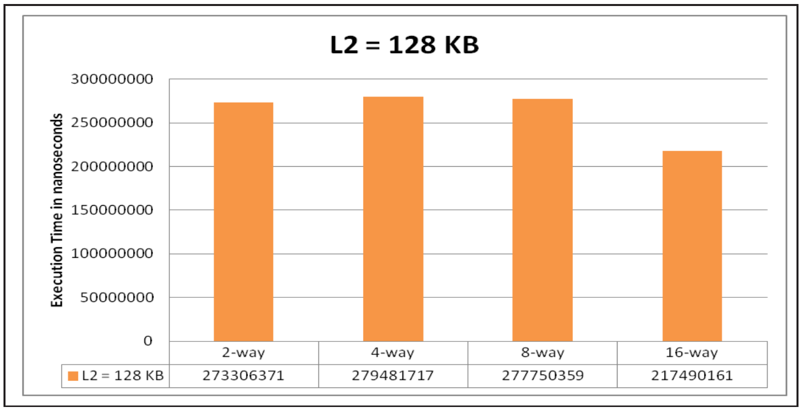

Figure 5.2: Execution time vs. set associativity for multicore mobile processor with 128 KB L2 cache

set associative L2 cache.

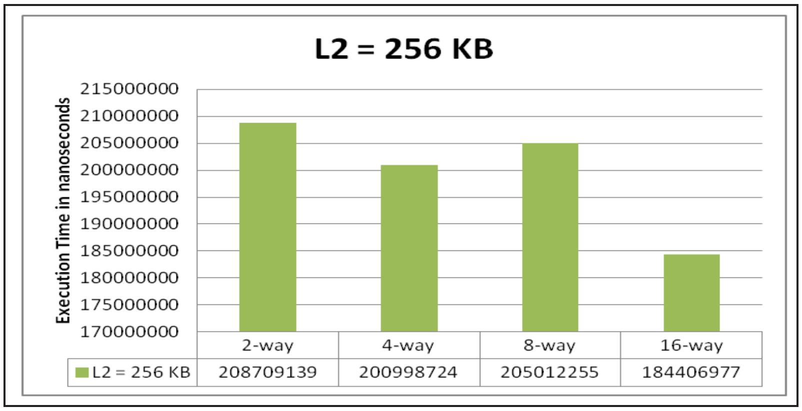

Figure 5.3: Execution time vs. set associativity for multicore mobile processor with 256 KB L2 cache

Figure 5.4, shows the execution time variation for FFT with a 2-way, 4-way, 8-way and 16-way set associative L2 cache of size 512 KB. For a multicore mobile processor with 2 cores, 16 KB L1 cache, and 512 KB L2 cache, the best performance can be observed for a 16-way set associative L2 cache.

Figure 5.5, shows the execution time of FFT in a 2-way, 4-way, 8-way and 16-way set associative L2 cache of size 1 MB. For a multicore mobile processor with 2 cores, 16 KB L1 cache, and 1 MB L2 cache, the best performance can be observed for a 8-way set associative L2 cache.

In each of the benchmark execution as the cache size increases the execution time

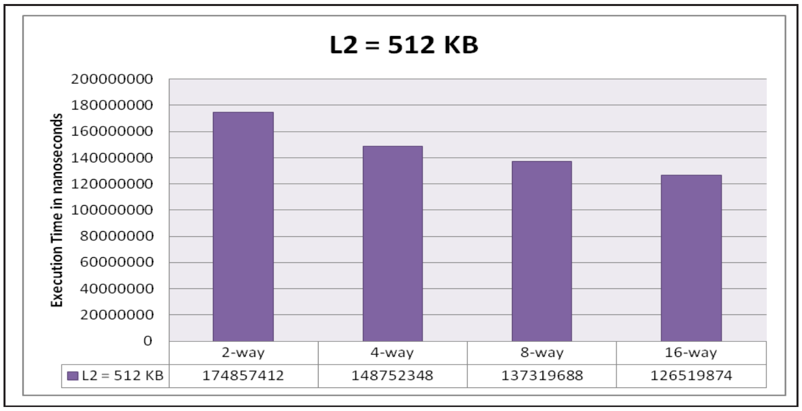

Figure 5.4: Execution time vs. set associativity for multicore mobile processor with 512 KB L2 cache

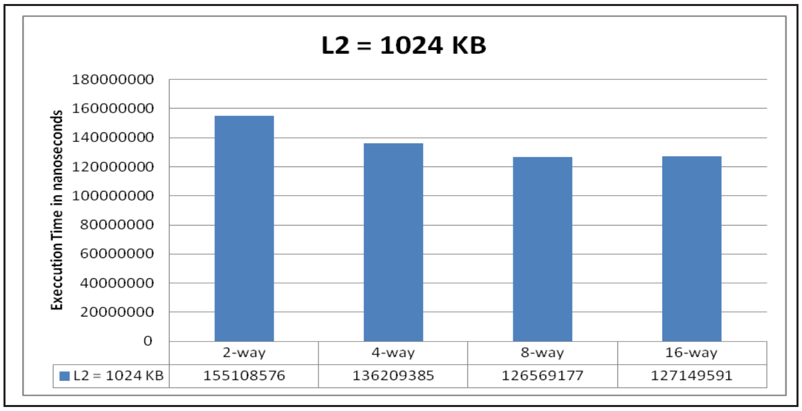

Figure 5.5: Execution time vs. set associativity for multicore mobile processor with 1024 KB L2 Cache

lowered. But it is observed that a 16-way, 512 KB L2 cache has lower execution time as compared to a 1 MB L2 cache with 2-way, 4-way, 8-way or 16-way set associativity.

5.5 Summary

A novel cache configuration, without increasing the cache size for performance improvement of multicore mobile processors is proposed in this chapter. Through extensive simulation we observed that the performance of a shared 16-way 512 KB L2 cache is better as compared to a 2-way, 4-way, 8-way or 16-way, 1 MB shared L2 cache. Thus, it can be concluded that a 16-way, 512 KB shared L2 cache is able to provide an optimal performance for multicore mobile processor.

Chapter 6

Conclusions and Future Works

The work in this thesis, primarily focuses on enhancement of performance for multicore processor. A novel interconnect for multicore processor has been proposed, which incorporates buffer memory and switches. The work reported in this thesis is summarized in this chapter. Section 6.1 deals with our contribution and Section 6.2 provides some scope for further development.

6.1 Conclusions

The thesis deals with performance enhancement using a new cache configuration for multicore processor, multicore processor with Interconnect network on chip and a new cache configuration for multicore mobile processors. The proposed cache configuration for multi-core processor provides better performance in comparison to existing cache configurations. The MPIN model with eight cores has been used for this performance analysis, which can be extended to cores of higher counts and processor architecture. A new interconnect structure INoC is proposed to reduce the communication delay in the core to memory or memory to memory communication. An optimal cache configuration for multicore mobile processor (8-way, 512 KB shared L2 cache) is proposed to enhance the performance.

6.2 Future Works

- In our proposed work, performance metrics like power, energy and temperature are not considered. The given cache configuration and novel interconnection mechanism can be extended and analysed to handle these above metrics.

- Interconnect network on chip is shown here with shared L2 cache, it can also be implemented with shared L3 cache
- The proposed network can be effectively used for heterogeneous processing cores as well.

Bibliography

[1] Aggarwal, N., Ranganathan, P., Jouppi, N. P., and Smith, J. E., "Configurable isolation: building high availability systems with commodity multi-core processors," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 470–481, 2007.

[2] Akhter, S. and Roberts, J., *Multi-core programming*, vol. 33. Intel Press, 2006.

[3] Balakrishnan, S., Rajwar, R., Upton, M., and Lai, K., "The impact of performance asymmetry in emerging multicore architectures," in *ACM SIGARCH Computer Architecture News*, vol. 33, pp. 506–517, IEEE Computer Society, 2005.

[4] Balasubramonian, R. and Pinkston, T. M., "Buses and crossbars," in *Encyclopedia of Parallel Computing*, pp. 200–205, Springer, 2011.

[5] Beavers, B., "The story behind the intel atom processor success," *IEEE Design & Test of Computers*, vol. 26, no. 2, pp. 8–13, 2009.

[6] Bhowmik, R. and Govindaraju, M., "L2 cache performance analysis and optimizations for processing hdf5 data on multi-core nodes," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pp. 142–149, IEEE, 2012.

[7] Bienia, C., Kumar, S., and Li, K., "PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors," in *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pp. 47–56, IEEE, 2008.

[8] Blake, G., Dreslinski, R. G., and Mudge, T., "A survey of multicore processors," *Signal Processing Magazine, IEEE*, vol. 26, no. 6, pp. 26–37, 2009.

[9] Blem, E., Esmaeilzadeh, H., St Amant, R., Sankaralingam, K., and Burger, D., "Multicore model from abstract single core inputs," 2012.

[10] Bournoutian, G. and Orailoglu, A., "Dynamic, multi-core cache coherence architecture for power-sensitive mobile processors," in *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2011 Proceedings of the 9th International Conference on*, pp. 89–97, IEEE, 2011.

[11] Brown, J. A., Kumar, R., and Tullsen, D., "Proximity-aware directory-based coherence for multi-core processor architectures," in *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pp. 126–134, ACM, 2007.

[12] Buckler, M., Burleson, W., and Sadowski, G., "Low-power networks-on-chip: Progress and remaining challenges," in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*, pp. 132–134, IEEE, 2013.

[13] Bui, J., Xu, C., and Gurumurthi, S., "Understanding Performance Issues on both Single Core and Multi-core Architecture," tech. rep., Technical report, University of Virginia, Department of Computer Science, Charlottesville, 2007.

[14] Chadwick, G. A., "Communication centric, multi-core, fine-grained processor," tech. rep., 2013.

[15] Chen, G., Hu, B., Huang, K., Knoll, A., and Liu, D., "Shared l2 cache management in multicore real-time system," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pp. 170–170, IEEE, 2014.

[16] Cheng, L., Carter, J. B., and Dai, D., "An adaptive cache coherence protocol optimized for producer-consumer sharing," in *High Performance Computer Archi-*

tecture, 2007. HPCA 2007. IEEE 13th International Symposium on, pp. 328–339, IEEE, 2007.

[17] Cragon, H. G., *Memory systems and pipelined processors*. Jones & Bartlett Learning, 1996.

[18] Creeger, M., "Multicore cpus for the masses," *Queue*, vol. 3, no. 7, pp. 64–ff, 2005.

[19] Dally, W. J., "Performance analysis of k-ary n-cube interconnection networks," *Computers, IEEE Transactions on*, vol. 39, no. 6, pp. 775–785, 1990.

[20] Dally, W. J. and Towles, B. P., *Principles and practices of interconnection networks*. Elsevier, 2004.

[21] Das, B., Mahato, A. K., and Khan, A. K., "Architecture and implementation issues of multi-core processors and caching–a survey," *IJRET*, vol. 2, 2013.

[22] Diavastos, A., Stylianou, G., and Trancoso, P., "Tfluxscc: a case study for exploiting performance in future many-core systems," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, p. 26, ACM, 2014.

[23] El-Moursy, A. A. and Sibai, F. N., "V-set cache design for llc of multi-core processors," in *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, pp. 995–1000, IEEE, 2012.

[24] Esmaeilzadeh, H., Blem, E., Amant, R. S., Sankaralingam, K., and Burger, D., "Power challenges may end the multicore era," *Communications of the ACM*, vol. 56, no. 2, pp. 93–102, 2013.

[25] Fasiku, A., Oyinloye, O., Falaki, S., and Adewale, O., "Performance evaluation of multicore processors," *International Journal of Engineering and Technology*, vol. 4, no. 1, 2014.

[26] FELLOWS, K. M., *A comparative study of the effects of parallelization on ARM and Intel based platforms*. PhD thesis, University of Illinois at Urbana–Champaign, 2014.

[27] Fide, S., *Architectural optimizations in multi-core processors*. VDM Verlag, 2008.

[28] Fruehe, J., "Multicore Processor Technology," pp. 67–72, 2005.

[29] Gal-On, S. and Levy, M., "Measuring multicore performance," *Computer*, vol. 41, no. 11, pp. 99–102, 2008.

[30] Geer, D., "Chip makers turn to multicore processors," *Computer*, vol. 38, no. 5, pp. 11–13, 2005.

[31] Geer, D., "For programmers, multicore chips mean multiple challenges," *Computer*, vol. 40, no. 9, pp. 17–19, 2007.

[32] Gepner, P. and Kowalik, M. F., "Multi-core processors: New way to achieve high system performance," in *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*, pp. 9–13, IEEE, 2006.

[33] Gillespie, M., "Preparing for the second stage of multi-core hardware: Asymmetric (heterogeneous) cores," *Intel white paper*, 2008.

[34] Gorder, P., "Multicore processors for science and engineering," *Computing in science & engineering*, vol. 9, no. 2, pp. 3–7, 2007.

[35] Goulding-Hotta, N., Sampson, J., Venkatesh, G., Garcia, S., Auricchio, J., Huang, P.-C., Arora, M., Nath, S., Bhatt, V., Babb, J., and others, "The greendroid mobile application processor: An architecture for silicon's dark future," *IEEE Micro*, vol. 31, no. 2, pp. 86–95, 2011.

[36] Gunn, C., "Cmos photonics for high-speed interconnects," *Micro, IEEE*, vol. 26, no. 2, pp. 58–66, 2006.

[37] Hackenberg, D., Molka, D., and Nagel, W. E., "Comparing cache architectures and coherency protocols on x86-64 multicore SMP systems," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on microarchitecture*, pp. 413–422, ACM, 2009.

[38] Hanada, T., Sasaki, H., Inoue, K., and Murakami, K., "Performance evaluation of 3d stacked multi-core processors with temperature consideration," in *3D Systems Integration Conference (3DIC), 2011 IEEE International*, pp. 1–5, IEEE, 2012.

[39] Hennessy, J. L. and Patterson, D. A., *Computer architecture: a quantitative approach*. Elsevier, 2012.

[40] Hijaz, F. and Khan, O., "Rethinking last-level cache management for multicores operating at near-threshold voltages," 2014.

[41] Home, D. J., "The effectiveness of multicore processors in modern smartphones," 2013.

[42] Huang, T., Zhu, Y., Qiu, M., Yin, X., and Wang, X., "Extending amdahls law and gustafsons law by evaluating interconnections on multi-core processors," *The Journal of Supercomputing*, pp. 1–15, 2013.

[43] Hughes, C. and Hughes, T., *Professional multicore programming: design and implementation for C++ developers*. Wrox, 2011.

[44] Hwang, K., *Advanced computer architecture*. Tata McGraw-Hill Education, 2003.

[45] Jouppi, N. P., "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *ACM SIGARCH Computer Architecture News*, vol. 18, pp. 364–373, ACM, 1990.

[46] Jun-Feng, B., "Application development methods based on multi-core systems," in *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pp. 858–862, IEEE, 2012.

[47] Kang, S., Arora, N., Shringarpure, A., Vuduc, R. W., and Bader, D. A., "Evaluating multicore processors and accelerators for dense numerical computations," *Multicore Computing: Algorithms, Architectures, and Applications*, p. 241, 2013.

[48] Khaitan, S. K. and McCalley, J. D., "Optimizing cache energy efficiency in multicore power system simulations," *Energy Systems*, vol. 5, no. 1, pp. 163–177, 2014.

[49] Khan, O., Lis, M., Sinangil, Y., and Devadas, S., "Dcc: A dependable cache coherence multicore architecture," *Computer Architecture Letters*, vol. 10, no. 1, pp. 12–15, 2011.

[50] Khan, S. M., Alameldeen, A. R., Wilkerson, C., Kulkarni, J., and Jimenez, D. A., "Improving multi-core performance using mixed-cell cache architecture," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pp. 119–130, IEEE, 2013.

[51] Kim, H., Maxwell, R., Patel, A., and Lee, B. K., *Performance Evaluation of Multi-threaded System Vs. Chip-multi Processor System*. PhD thesis, University of Texas at San Antonio, 2012.

[52] Kirner, R. and Puschner, P., "Obstacles in worst-case execution time analysis," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pp. 333–339, IEEE, 2008.

[53] Knight, W., "Two heads are better than one [dual-core processors]," *IEE Review*, vol. 51, no. 9, pp. 32–35, 2005.

[54] Kumar, R., Tullsen, D., Ranganathan, P., Jouppi, N., and Farkas, K., "Single-ISA heterogeneous multi-core architectures for multithreaded workload performance," in *ACM SIGARCH Computer Architecture News*, vol. 32, p. 64, IEEE Computer Society, 2004.

[55] Kumar, R., Zyuban, V., and Tullsen, D. M., "Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling," in *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on*, pp. 408–419, IEEE, 2005.

[56] McDaniel, M. and Dorn, J., "Intel atom processor," *University of Virginia*, pp. 1–16, 2009.

[57] Michaud, P., "Demystifying multicore throughput metrics," *IEEE Computer Architecture Letters*, 2012.

[58] Molka, D., Hackenberg, D., Schone, R., and Muller, M. S., "Memory performance and cache coherency effects on an intel nehalem multiprocessor system," in *Parallel Architectures and Compilation Techniques, 2009. PACT'09. 18th International Conference on*, pp. 261–270, IEEE, 2009.

[59] Monchiero, M., Canal, R., and González, A., "Design space exploration for multicore architectures: a power/performance/thermal view," in *Proceedings of the 20th annual international conference on Supercomputing*, pp. 177–186, ACM, 2006.

[60] Monchiero, M., Canal, R., and Gonzalez, A., "Power/performance/thermal design-space exploration for multicore architectures," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 5, pp. 666–681, 2008.

[61] Moore, G. E., "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff.," *Solid-State Circuits Newsletter, IEEE*, vol. 11, no. 5, pp. 33–35, 2006.

[62] Nayfeh, B. and Olukotun, K., "A single-chip multiprocessor," *Computer*, vol. 30, no. 9, pp. 79–85, 1997.

[63] Olson, D., "Intel Announces Plan for up to 8-core Processor," *Slippery Brick, March*, 2008.

[64] Owens, J. D., Dally, W. J., Ho, R., Jayasimha, D., Keckler, S. W., and Peh, L.-S., "Research challenges for on-chip interconnection networks," *Micro, IEEE*, vol. 27, no. 5, pp. 96–108, 2007.

[65] Parhami, B., *Computer arithmetic: algorithms and hardware designs*. Oxford University Press, Inc., 2009.

[66] Peng, L., Peir, J., Prakash, T., Chen, Y., and Koppelman, D., "Memory performance and scalability of Intel's and AMD's dual-core processors: a case study," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, pp. 55–64, IEEE, 2007.

[67] Pham, D., Asano, S., Bolliger, M., Day, M., Hofstee, H., Johns, C., Kahle, J., Kameyama, A., Keaty, J., Masubuchi, Y., and others, "The design and implementation of a first-generation CELL processor," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, pp. 184–592, IEEE, 2005.

[68] Portero, A., Scionti, A., Yu, Z., Faraboschi, P., Concatto, C., Carro, L., Garbade, A., Weis, S., Ungerer, T., and Giorgi, R., "Simulating the future kilo-x86-64 core processors and their infrastructure," in *Proceedings of the 45th Annual Simulation Symposium*, p. 9, Society for Computer Simulation International, 2012.

[69] Ramasubramanian, N., Gounden, N. A., and others, "Performance of cache memory subsystems for multicore architectures," *arXiv preprint arXiv:1111.3056*, 2011.

[70] Roberts-Hoffman, K. and Hegde, P., "Arm cortex-a8 vs. intel atom: Architectural and benchmark comparisons," *Dallas: University of Texas at Dallas*, 2009.

[71] Schauer, B., "Multicore Processors–A Necessity," *ProQuest Discovery Guides1–14*, 2008.

[72] Shacham, A., Bergman, K., and Carloni, L. P., "Photonic networks-on-chip for future generations of chip multiprocessors," *Computers, IEEE Transactions on*, vol. 57, no. 9, pp. 1246–1260, 2008.

[73] Smith, A. J., "Line (block) size choice for cpu cache memories," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 1063–1075, 1987.

[74] Sui, X., Wu, J., Chen, G., Tang, Y., and Zhu, X., "Augmenting cache partitioning with thread-aware insertion/promotion policies to manage shared caches," in *Proceedings of the 7th ACM international conference on Computing frontiers*, pp. 79–80, ACM, 2010.

[75] Sweazey, P. and Smith, A. J., "A class of compatible cache consistency protocols and their support by the IEEE futurebus," in *ACM SIGARCH Computer Architecture News*, vol. 14, pp. 414–423, IEEE Computer Society Press, 1986.

[76] Tullsen, D. M., Eggers, S. J., Emer, J. S., Levy, H. M., Lo, J. L., and Stamm, R. L., "Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor," in *ACM SIGARCH Computer Architecture News*, vol. 24, pp. 191–202, ACM, 1996.

[77] Tullsen, D. M., Eggers, S. J., and Levy, H. M., "Simultaneous multithreading: maximizing on-chip parallelism," in *25 years of the international symposia on Computer architecture (selected papers)*, pp. 533–544, ACM, 1998.

[78] Ubal, R., Jang, B., Mistry, P., Schaa, D., and Kaeli, D., "Multi2Sim: a simulation framework for CPU-GPU computing," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, pp. 335–344, ACM, 2012.

[79] Ubal, R., Sahuquillo, J., Petit, S., and López, P., "Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithread Processors," in *IEEE 19th International Symposium on Computer Architecture and High Performance computing, page (s)*, pp. 62–68, 2007.

[80] Ubal, R., Sahuquillo, J., Petit, S., López, P., Chen, Z., and Kaeli, D., "The Multi2Sim Simulation Framework."

[81] Wang, R., Gao, Y., and Zhang, G., "Real time cache performance analyzing for multi-core parallel programs," in *Cloud and Service Computing (CSC), 2013 International Conference on*, pp. 16–23, IEEE, 2013.

[82] Wang, X., Gan, G., Manzano, J., Fan, D., and Guo, S., "A quantitative study of the on-chip network and memory hierarchy design for many-core processor," in *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, pp. 689–696, IEEE, 2008.

[83] Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., and Gupta, A., "The SPLASH-2 programs: Characterization and methodological considerations," in *ACM SIGARCH Computer Architecture News*, vol. 23, pp. 24–36, ACM, 1995.

[84] Wu, H., *An Express Network-on-Chip (ExNoC) Cache Architecture for Large Caches*. PhD thesis, University of Cincinnati, 2011.

[85] Wu, J., Sui, X., Tang, Y., Zhu, X., Wang, J., and Chen, G., "Cache management with partitioning-aware eviction and thread-aware insertion/promotion policy," in *Parallel and Distributed Processing with Applications (ISPA), 2010 International Symposium on*, pp. 374–381, IEEE, 2010.

[86] Xie, Y., *Efficient shared cache management in multicore processors*. PhD thesis, Georgia Institute of Technology, 2011.

[87] Xu, T. C., Liljeberg, P., and Tenhunen, H., "Explorations of optimal core and cache placements for chip multiprocessor," in *NORCHIP, 2011*, pp. 1–6, IEEE, 2011.

[88] Zhou, H., Bhopalwala, M., and Roveda, J., "On-chip interconnect network communication management for multi-core design," *Journal of Advances in Computer Network*, vol. 1, no. 3, 2013.