Analysing the Performance of Multi-core Architecture

Ram Prasad Mohanty [#], Ashok Kumar Turuk [*], Bibhudatta Sahoo [#]

[#] *Computer Science Department, National Institute of Technology Rourkela*

ABSTRACT
The advancement in technology has brought immense amount of changes in the design and productivity of applications designed for being used in the personal computers. By implementing greater number of cores to the same chip also results in facing challenges. In this case the challenge that is being faced is the core to core communication as well as the memory in addition to cache coherence. This paper presents a detailed analysis on performance of FFT a divide and conquer algorithm across with the Multi-core architecture with Internal and external network. The architectures are being defined using memory configuration and context configuration with help of Multi2Sim 3.4 simulator. The performance of these architectures have been simulated with Splash 2 Benchmark.

Keywords
Multi-core Technology, Multi-core Issues, SPLASH2 Benchmark, performance, Multi2Sim simulator

1. Introduction

In the previous decade enhancement in the processor speed were done on a high basis but still the requirement was not achieved. New approach was essential by the computer architecture so that they can provide adequate enhancement in the performance. It was predicted that by placing an extra processing core in the same chip, there shall be enhancement in the performance, as well as lower production of heat, but the actual speed of the core was lower in comparison to the single core processor. The IEEE reviewed in September 2005 that the power consumption increases up to 60% with the use of every 400 MHz rise in clock speed, it also cited that we can get considerable improvement in performance through the means of dual-core approach [4].

The concept of multi-core is not an innovative one; the idea is used in various systems, and for some period of time it has also been used for specialized applications. But currently this technology has become extremely conventional with Intel and Advanced Micro Devices (AMD) developing many commercially accessible multi-core chips. In the year 2008, two and four core machines were commercially accessible. Some experts are of belief that by the year 2017, 4,096 cores would be supported by the embedded processors, 512 cores might be upheld by the server CPUs and 128 cores could be used by the desktop [11]. In past 30 years the desktop chips used a single core, but today the desktop chips use four cores, this shows that the rate of growth is really amazing.

In the multi-core processor technology CMP that is Chip Multiprocessing is used. Execution cores have their individual set of execution and architectural resources. The different processor architecture is given in Fig: 1 [26].

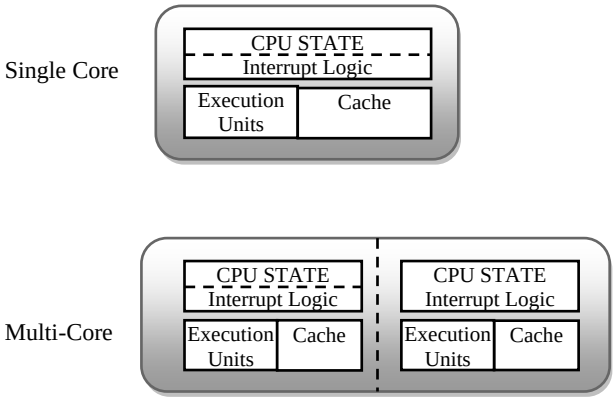

Fig. 1 Comparison of Multi-Core and Single Core Architecture

At the initial stage in order to enhance the performance a very simple tractable method was used and that was to increase or enhance the frequency of the processor. Thus from generation to generation tracking the performance of the processor was a very easy task. But as the frequencies rose higher and higher at a particular stage a reality came into picture that leads to other advancements. With higher frequency the dissipation of heat enhanced. At the same time the power consumption also increased. Thus the concept of implementing multiple cores onto the processor was developed. This leads to a solution to the heat and power issue. But it also leads to new issues and much more interesting problematic areas.

This paper uses a simulation approach to study the performance on two different multicore architectures using multi2Sim. The next section discusses on various performances related issues with multi-core processor architecture. Section 3 describes the different multicore architecture with processor core, cache, switch and main memory. Simulation frame work multi2Sim is discussed with configuration details for two different multicore architecture (i) Multi-core processor using Internal Networks and (ii) Multi-core processor using external Networks are presented in Section 4. Finally, conclusions and directions for future research are discussed in Section 5.

2. MULTI-CORE CHALLENGES

Certain problems and dispute come forward when the multiple cores are implemented and set upon a single chip. Power and temperature supervision is an enormous disquiet that leads to an exponential rise with the accumulation of additional cores. Another dispute that occurs in the multi-core is the memory inconsistency. There will be no benefit if the programmers do not inscribe applications that acquire the benefit of multi-core. It is very essential that application should be written so that varied part can run simultaneously. Seven of the issues are described below.

A. Issues occurred because of handling of power and temperature

With excessive power consumption there exists immense amount of heat dissipation. In the same way if the chip used for implementing a single chip holds two or more cores on it without any kind of updation made to the chip. In such case it may lead to consumption of double power and likewise generation of even larger amount of heat. Also in the extreme conditions this can lead to combustion of the computer. In order to avoid such cases the individual cores are executed at lower frequencies. Even in the current trend every design integrate a power control unit which is so designed that it can go ahead to stop or shut down the cores that remains unutilized thus restricting the power consumption.

Heat generation is taken care by restricting the quantity of hot spots over the chip. This too is handled during the design level. The design is so made that the hot spots does not grow too high in numbers and at the same time the heat generated are spread across the chip.

B. Issues due to cache coherence

One of the most important issues still remaining a prime concern in the multi-core environment is the distribution of different types of caches across the chip. The L1 and the L2 cache which is distributed across the chip prove to be the prime concern in the environment of multi-core. When each core has its own individual cache then the value at each cache may not hold the updated values or the actual required values.

Two types of protocols are used in general for handling cache coherence. Snooping protocol is a protocol which can only come into usage in systems based on bus architecture. It takes the aid of a number of states. Using these states it can determine which values in the cache needs to be updated. The snooping protocol is not at all scalable. The other protocol is directory based which is highly scalable. This protocol can easily be used on an arbitrary network thus it can easily be scaled to multiple processors or cores.

C. Multithreading

The major problem in using multithreading is to acquire great performance through the multi-core processor. Reconstruction of the application to be multithreading indicates that the programmers have to revise in most of the cases. The application is to be written by the programmers with the subroutine capable to perform on various cores. This signifies that the data dependence ought to be handled in a very synchronized and structured way. The programmers are not acquiring the benefit of the multi-core system if a particular core is used more than the other. Few companies have manufactured a new product with the capability of the multi-core. The recent operating device, produced by the Microsoft and Apple can run up to 4 cores.

D. Requirement of Enhancement in the memory system

It is very essential to increase the memory, when numbers of cores are placed on a single chip. The Pentium 4 processor which is a 32 bit processor has the capacity of addressing main memory up to a limit of 4 GB. At the current juncture there exists 64 bit processor as well which can handle infinite amount of addressable memory. Thus it becomes highly necessary to develop an enhanced memory system that can handle this amount. At the same time higher amount of main memory as well as even larger caches are required for the current multithreaded multiprocessors.

E. Requirement of Enhancement in the in the interconnection networks as well as the system bus network

Even if the amount of main memory gets enhanced still without a proper management of the time required to handle memory request the benefit is not utilized. These days the interconnection network that exists between the cores has become the prime concern of the manufacturers. When the network gets faster the latency reduces in the communication between cores as well as memory transactions. Recently Intel has come up with its Quickpath Interconnect. It provides point-to-point links at both sides of the processor which is of high-speed. The speed of transfer is enhanced because of the connection between the distributed shared memory, I/O hub, Intel processors as well as the internal cores. AMD also developed the hyper Transport technology which is a wide bus based system. Similarly a new interconnect could be 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 multi-core processors.

F. Need of Programming in Parallel Environment

In May 2007, an employee of Intel, Shekhar Borkar mentioned that the Moore's Law has also been followed by the software development. The quantity of parallelism should be doubled during software development, so that every two years it can be able to sustain even in the fast advancing multi-core architecture [5]. As the total amount of core present in a processor has to be twofold in every 18 months as per Moore's law. Thus the programmers need to learn how to write programs in such a way that they can be divided and run parallel on the multiple cores instead of using the single core hardware only. The main reason of parallel programming is enhance the productivity of the multi-core processors.

Some hidden concerns are being bought up in the software being developed to be executed in the environment of multi processors. The first question that proves to be a big concern is how a programmer provides priority information to tasks which have higher priority over other tasks in the queue? This priority is not restricted to a single core instead the priority needs to be established throughout the processor. As per the design of the thread, even if it has been given the high priority within the level of the core, it is not necessary that it shall get the highest priority all over the system. It is also a hard task to know whether the whole system is stopped to function or only the core on which the application is running is stopped.

These problems should be solved while teaching the developers the best practices for parallel programming. It would be quite easy to be on track with the Moore's law concept if the programmers have the basic idea of the multithreading and also have the power of programming in the parallel environment.

G. Cores are not getting the data

One or many cores may be remaining idle by waiting for the data, if a program has not been developed properly to utilize the cores of a multiprocessor. This can be visible if the multi-core system is used to run a single-threaded application. In such case the thread will function on a single core while the other cores will be have no function to operate on? Still these cores keep on making calls to the main memory which utilizes a loss of clock cycles. This shall add on to the penalty thus reducing the overall performance. Thus a proper replacement policy need to be utilized which can lead to removal of all the cache entries that has been processed by the other cores. With

the addition of number of cores in the processor, this problem gets more deep and troublesome.

3. MULTI-CORE ARCHITECTURES

This section outlines the basic details of four different multi-core architectures (i) Multi-core processor with Internal Network, (ii) Multi-core Processor with External Network, (iii) Mul;ti-core processor with Ring Network and (iv) Heterogeneous system with CPU and GPU cores [22].

Few of the Multi-Core Architecture which can be used for a thorough performance analysis of the divide and conquer algorithms on Multi-core Processor with Internal Network as described below.

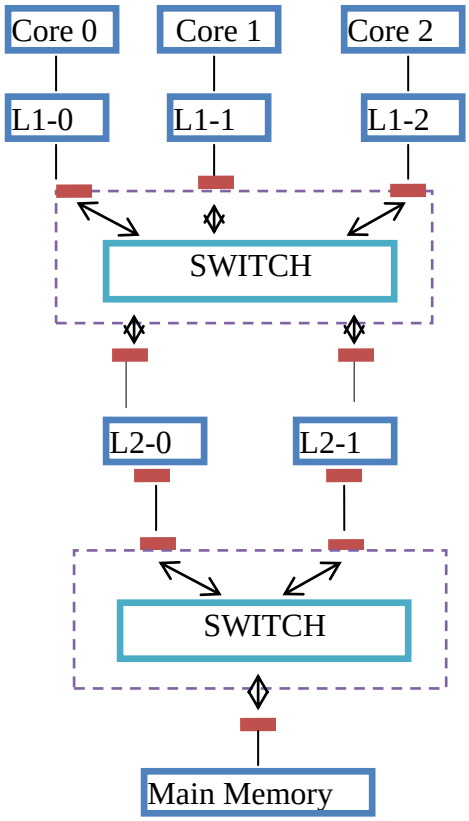

Fig. 2 Multi-Core Processor using Internal Network

In the Fig. 2 the architecture of a multi-core processor with internal network has been shown that has a single private L1 cache per core, to unify the instruction and data requests. Two L2 caches are being used to share between all the three cores.

Multi-core processor with External Network is shown in Fig.3. The connectivity of L1-to-L2 network consists of two distinct switches and 5 nodes (n0, n1, n2, n3, n4) [22], which further communicate with main memory through another switch.

The third architecture of multi-core processor is complex and uses ring network to connect main memory modules with multiple cores. Typical 4 core multi-core processor architecture is shown in Fig 4. This architecture is a more complex architecture than the above two. It uses a 4-core processor which have private L1 data caches and a common L1 instruction cache shared by every two cores as shown in Fig 4

The two L2 caches serve the independent higher level L1 caches.

The computing capability of multicore processor can be further optimized with the help of CPU and GPU integration. A heterogeneous system which has CPU and GPU cores as shown in Fig: 5

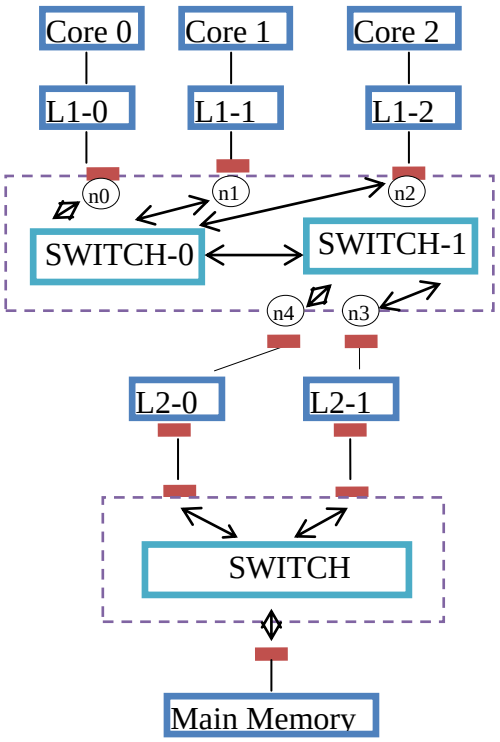

Fig. 3 Multi-Core processor using External Networks

This architecture uses CPU and GPU cores as well as it has a single CPU core formed of two hardware threads which is used together with a GPU having 4 compute units. Each CPU thread has a private L1 cache, while one L1 cache is shared with every two GPU compute units [22]. We have analyzed the performance of two architectures (i) Multi-Core Architecture with internal network, and (ii) Multi-Core Architecture with external network using multi2Sim simulator.

4. Experimental results

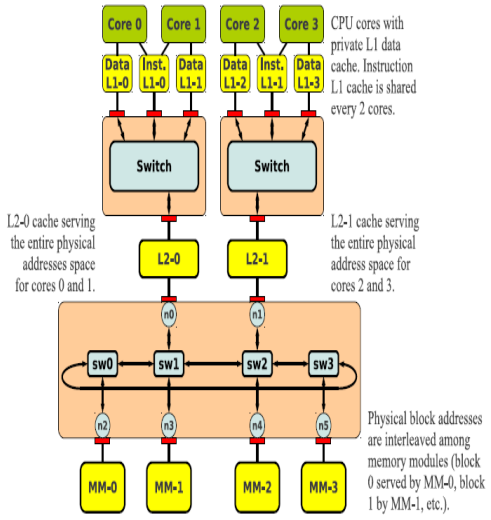

Fig. 4 Multi-Core processor using Ring Network [22]

Four major steps were involved in our research:

1) Setting up the environment on which the code gets executed

2) Setting up the code that the simulator will execute

3) Execute single-core simulations (Collect and Analyze Data)

4) Execute multi-core simulations (Collect and Analyze Data)

For the first step it was decided to use the multi2Sim simulator because of its versatility of implementing multi-core architectures. Also we had to fix a commonly used benchmark.

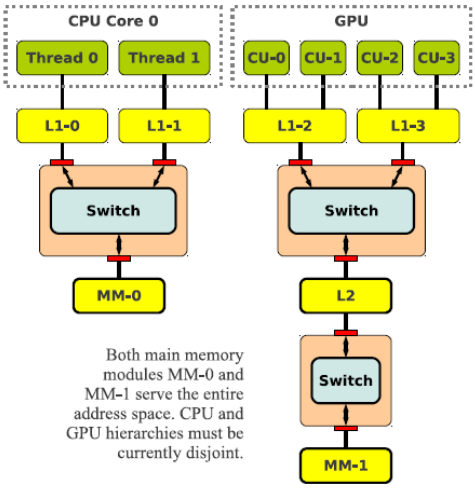

Fig. 5 Heterogeneous system with CPU and GPU cores [22]

The single core simulations were also executed. The following factors were changed: L1 Cache size, L2 Cache size, and Bus width, Latency, and cache coherence protocols. Instead of varying the values for each factor repeatedly, we decided the best possible values for each and then proceeded with the execution of the code.

A. Simulation Environment

With the advancement of processor architecture over time, benchmarks that were used to compute the performance of these processors are not as practical today as they were before due to their incapability to stress the new architectures to their utmost capacity in terms of clock cycles, cache, main memory and I/O bandwidth.

Hence new and enhanced benchmarks have to be developed and used. The SPLASH-2 is one such benchmark that has concentrated workloads based on real applications and is a descendant of the SPLASH benchmark. Other such benchmark includes PARSEC, CPUSPEC2006, and Mini Bench. The SPLASH-2 has 11 programs. The details of these benchmark programs are shown in Tables 2.1 [23, 24]. All experiments were run on systems with 32 bit LINUX operating system and Intel Core 2 Duo processors using the multi2Sim simulator.

TABLE I Specification of SPLASH2 Benchmark

Sl No.	Benchmark	Application Domain	Problem Size
1	Barnes	High-Performance Computing	65536 Particles
2	Cholesky	High-performance computing	Tk29.0
3	FFT	Signal Processing	4,194,304 data points
4	FMM	High-Performance Computing	65536 Particles
5	LU	High-Performance Computing	1024X1024 matrix, 64X64 Blocks
6	Ocean	High-Performance Computing	514X514 Grid
7	RADIOSITY	Graphics	Large room
8	RADIX	General	8388608 integers
9	RAYTRACE	Graphics	Car
10	WATER-NSQUARED	High-Performance Computing	4096 molecules
11	WATER-SPATIAL	High-Performance Computing	4096 molecules

The Multi2Sim 3.4 is used to conduct the experiment that uses gcc, glib, freeglut and gtk+ packages.

B. Performance analysis of Multicore Architecture with Internal Network

We analyzed the performance of the code across the multi-core architectures with internal networks as well as with external networks. The detailed architecture for the same has been provided in the figure 3 :

The cache specification for the same architecture is shown as below [22]:

```
[CacheGeometry geo-l1]
Sets = 128
Assoc = 2
BlockSize = 256
Latency = 2
Policy = LRU
Ports = 2
[CacheGeometry geo-l2]
Sets = 512
Assoc = 4
BlockSize = 256
Latency = 20
Policy = LRU
Ports = 4
[Module mod-l1-0]
Type = Cache
Geometry = geo-l1
LowNetwork = net-l1-l2
LowModules = mod-l2-0 mod-l2-1
[Module mod-l1-1]
Type = Cache
Geometry = geo-l1
LowNetwork = net-l1-l2
LowModules = mod-l2-0 mod-l2-1
[Module mod-l1-2]
Type = Cache
Geometry = geo-l1
LowNetwork = net-l1-l2
LowModules = mod-l2-0 mod-l2-1
[Module mod-l2-0]
Type = Cache
Geometry = geo-l2
HighNetwork = net-l1-l2
LowNetwork = net-l2-mm
LowModules = mod-l2-mm
AddressRange = BOUNDS 0x00000000 0x7FFFFFFF
```

```
[Network net-l1-l2]
DefaultInputBufferSize = 1024
DefaultOutputBufferSize = 1024
DefaultBandwidth = 256
[Network net-l2-mm]
DefaultInputBufferSize = 1024
DefaultOutputBufferSize = 1024
DefaultBandwidth = 256
[Module mod-l2-1]
Type = Cache
Geometry = geo-l2
HighNetwork = net-l1-l2
LowNetwork = net-l2-mm
LowModules = mod-l2-mm
AddressRange = BOUNDS 0x80000000 0xFFFFFFFF
[Module mod-mm]
Type = MainMemory
BlockSize = 256
Latency = 200
HighNetwork = net-l2-mm
[Entry core-0]
Type = CPU
Core = 0
Thread = 0
DataModule = mod-l1-0
InstModule = mod-l1-0
[Entry core-1]
Type = CPU
Core = 1
Thread = 0
DataModule = mod-l1-1
InstModule = mod-l1-1
[Entry core-2]
Type = CPU
Core = 2
Thread = 0
DataModule = mod-l1-2
InstModule = mod-l1-2
```

C. Performance of Multicore Architecture with External Network

In the Fig 4 architecture of a multi-core processor with external network has been shown.

In this architecture each of the cores is associated with a L1 and L2 cache module. The nodes which are connected to the L1 cache are connected to the switch 0 while the ones connected to the L2 cache is connected to another switch.

The cache configuration for the given setup is given as below:

```
[CacheGeometry geo-l1]
Sets = 128
Assoc = 2
BlockSize = 256
Latency = 2
Policy = LRU
Ports = 2
[CacheGeometry geo-l2]
Sets = 512
Assoc = 4
BlockSize = 256
Latency = 20
Policy = LRU
Ports = 4
[Module mod-l1-0]
Type = Cache
Geometry = geo-l1
LowNetwork = net0
LowNetworkNode = n0
LowModules = mod-l2-0 mod-l2-1
[Module mod-l1-1]
Type = Cache
Geometry = geo-l1
LowNetwork = net0
LowNetworkNode = n1
LowModules = mod-l2-0 mod-l2-1
[Module mod-l1-2]
Type = Cache
Geometry = geo-l1
LowNetwork = net0
LowNetworkNode = n2
LowModules = mod-l2-0 mod-l2-1
[Module mod-l2-0]
Type = Cache
Geometry = geo-l2
HighNetwork = net0
HighNetworkNode = n3
LowNetwork = net-l2-mm
AddressRange = BOUNDS 0x00000000 0x7FFFFFFF
LowModules = mod-mm
```

```
[Module mod-l2-1]
Type = Cache
Geometry = geo-l2
HighNetwork = net0
HighNetworkNode = n4
LowNetwork = net-l2-mm
AddressRange = BOUNDS 0x80000000 0xFFFFFFFF
LowModules = mod-mm
[Module mod-mm]
Type = MainMemory
BlockSize = 256
Latency = 100
HighNetwork = net-l2-mm
[Network net-l2-mm]
DefaultInputBufferSize = 1024
DefaultOutputBufferSize = 1024
DefaultBandwidth = 256
[Entry core-0]
Type = CPU
Core = 0
Thread = 0
DataModule = mod-l1-0
InstModule = mod-l1-0
[Entry core-1]
Type = CPU
Core = 1
Thread = 0
DataModule = mod-l1-1
InstModule = mod-l1-1
[Entry core-2]
Type = CPU
Core = 2
Thread = 0
DataModule = mod-l1-2
InstModule = mod-l1-2
```

The network configuration set for this external network is as shown below:

```
[Network.net0]
DefaultInputBufferSize = 1024
DefaultOutputBufferSize = 1024
DefaultBandwidth = 256
[Network.net0.Node.sw0]
Type = Switch
[Network.net0.Node.n0]
Type = EndNode
[Network.net0.Node.n1]
Type = EndNode
[Network.net0.Node.n2]
Type = EndNode
[Network.net0.Node.sw1]
Type = Switch
[Network.net0.Node.n3]
Type = EndNode
[Network.net0.Node.n4]
Type = EndNode
[Network.net0.Link.sw0-n0]
Source = sw0
Dest = n0
Type = Bidirectional
[Network.net0.Link.sw0-n1]
Source = sw0
Dest = n1
Type = Bidirectional
[Network.net0.Link.sw0-n2]
Source = sw0
Dest = n2
Type = Bidirectional
[Network.net0.Link.sw0-sw1]
Source = sw0
Dest = sw1
Type = Bidirectional
[Network.net0.Link.sw1-n3]
Source = sw1
Dest = n3
Type = Bidirectional
[Network.net0.Link.sw1-n4]
Source = sw1
Dest = n4
Type = Bidirectional
```

5. CONCLUSIONS AND FUTURE WORK

The Multi-core processors are developed to enhance performance of computing. The utilization of the processor can be 100% only when the applications being executed is multithreaded. Only a few applications exists which are multithreaded and can be executed parallel. At the same time only a few programmers have the idea and intellect to write programs that can utilize the multi-core processor properly. This study helps us to select appropriate cores in a processor, cache organization/configuration and interconnect network. .

The same program shall be executed on the ring network and heterogeneous system with CPU and GPU cores and a thorough analysis of the performance shall be gathered.

6. REFERENCES

[1] Cameron Hughes and Tracey Hughes,"Professional Multi-core Programming",Wiley Publishing,2009

[2] Darryl Gove, " Multi-core Application Programming", Pearson, 2011

[3] Julian Bui and Chegguang Xu and Sudhanva Gurumurthi, "Understanding performance issues on both single core and multi-core Architecture", Computer Organization , 2007

[4] John Freuhe, "Planning Considerations for Multi-core Processor Technology", Dell Power Solutions, May 2005.

[5] P. Frost Gorder, "Multi-core Processors for Science and Engineering", IEEE CS, March/April 2007.

[6] L. Peng et al, "Memory Performance and Scalability of Intel‟s and AMD‟s Dual-Core Processors: A Case Study", IEEE, 2007.

[7] D. Geer, "Chip Makers Turn to Multi-core Processors", Computer, IEEE Computer Society, May 2005.

[8] D. Pham et al, "The Design and Implementation of a First-Generation CELL Processor", ISSCC.

[9] R. Goering, "Panel Confronts Multi-core Pros and Cons", [Online]. Available: http://www.eetimes.com/news/design/showArticle.jhtml?articleID=183702416

[10] R. Merritt, "CPU Designers Debate Multi-core Future", EETimes Online, February 2008, [Online]. Available: http://www.eetimes.com/showArticle.jhtml?articleID=206105179

[11] *Bryan Schauer, "Multi-core Processors – A Necessity", ProQuest, September 2008.*

[12] Bai Jun-Feng, "Application Development Methods Based On Multi-core Systems", American Journal of Engineering and Technology Research", 2011.

[13] S. Balakrishnan et al, "The Impact of Performance Asymmetry in Emergng Multi-core Architectures", Proceedings of the 32nd International Symposium on Computer Architecture, 2005.

[14] D. Geer, "For Programmers, Multi-core Chips Mean Multiple Challenges", Computer, September 2007.

[15] *B. Brey, "The Intel Microprocessors", Sixth Edition, Prentice Hall, 2003.*

[16] D. Stasiak et al, "Cell Processor Low-Power Design Methodology", IEEE Micro, September 2005

[17] W.Knight, "Two Heads are better than One", IEEE Review, September 2005

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

[19] S.Mukherjee and M. Hill, "Using Prediction to accelerate Coherence Protocols", ISCA, 1998.

[20] R.Kumar et al, "Single-ISA Heterogeneous Multi-core Architectures with Multithreaded Workload Performance", ISCA, June 2004.

[21] Zhongliang Chan et al, "The Multi2Sim Simulation Framework"

[22] R. Ubal and J. Sahuquillo and S. Petit and P. L\'opez, " Multi2Sim: A Simulation Framework to Evaluate Multi-core-Multithreaded Processors",Proc. of the 19th Int'l Symposium on Computer Architecture and High Performance Computin,Oct, 2007

[23] S.C. Woo and M. Ohara and E. Torrie and J. P. Singh and A. Gupta, " The SPLASH-2 Programs: Characterization and Methodological Considerations",Proc. of the 22nd Int'l Symposium on Computer Architecture, June,1995.

[24] Tribuvan Kumar Prakash, " PERFORMANCE ANALYSIS OF INTEL CORE 2 DUO PROCESSOR",BioPerf, August,2007.

[25] Christian Bienia and Sanjeev Kumar and Kai Li, " PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors", Princeton publications, September, 2008

[26] Shameem Akhter and Jason Roberts, " Increasing Performance through Software Multi-threading", Intel Press, 2006