HARDBACK SOLUTION TO ACCELERATE MULTIMEDIA COMPUTATION THROUGH MGP IN CMP

Jimcy Babu[1], Kavitha .V[2], K.V.Ramakrishnan[3]

[1]M Tech, CMRIT, Bangalore
[2]Ph D Scholar, Jain University
[3]Professor

Abstract

Current multimedia applications are becoming more and more complex; thereby it increases workload for General purpose processors (GPPs). This resulted in the advent of Chip multiprocessors (CMPs). In most of the cases CMPs were not fully utilized. Hence, how to achieve the contemporary multimedia requirement like speed i.e. completion time, by using CMP became a question.

In this project work a solution had been put forward to tackle the crisis. The idea behind the solution is the proper exploitation of the parallelism thereby accelerating the multimedia computation in CMPs. To achieve the less completion time, four levels of parallelism has been considered i.e. Data Level, Thread Level, Instruction Level and Memory Level called as Multi-Grain Parallelism (MGP). POSIX thread concept is used to implement Data Level, Thread Level, Instruction Level and for Memory Level, pre-fetch concept. To run, Linux based platform is required. Here UBUNTU 12.02 version is used.

This project also performs a comparison between serial computation and parallel computation at different levels, based on completion time. The experimental results show that parallel computation consumes less time as compared to that of serial computation. Thereby, making it viable for multimedia applications

Keywords: *Chip Multiprocessor (CMP), Data Level Parallelism (DLP), Thread Level Parallelism (TLP), Instruction Level Parallelism (ILP), Memory Level Parallelism (MLP), POSIX thread (p-thread), Pre-fetch.*

----------***----------

1. INTRODUCTION

Advances in IC technology have led to billions of transistors on chip keeping up with Moore's law. The initial trend was of CPU's with wider instruction issue and instruction execution involving prediction and speculation. This was the superscalar approach which was argued against due to its diminishing performance for increasing issue width due to limited amount of parallelism in instructions in non-scientific applications and the complex hardware needed. Many factors both technological and marketing are driving the semiconductor industry to implement multiple processors per chip.

Hence instead of a complex superscalar processor, an alternate approach, the Chip Multiprocessor (CMP) or multi-core processor was proposed. The CMP combines much simpler processors or cores on a single chip or die. Each core is a complete processor unit itself and works as a team with the other cores on chip. It is now proven that this approach is the only way to build high performance architectures and CMPs perform equal or better than the superscalar approach.

Chip Multiprocessor- also called Multi-core microprocessors or CMP for short are now the only way to build high performance microprocessors, for a variety of reasons. Large uniprocessor are no longer scaling in performance, because it is only possible to extract a limited amount of parallelism from a typical instruction stream using conventional superscalar instruction issue techniques. Along with the numerous opportunities of the CMPs, there are also a lot of challenges that keep us from exploiting their full potential. Numerous bottlenecks have appeared that have to be dealt with before we can fully benefit from CMPs. These bottlenecks have tapered off the performance increase of CMPs in recent years [4].

The rest of the paper is organised as follows. Section 2 analyses the related work carried. Section 3 comprises of methods for the parallel programming. Section 4, describes the simulation result

2. RELATED WORK

For real-time multimedia applications, performance is the key constraint. A fair comparison of energy must therefore also consider performance. As a result, the energy of SMT and CMP at the same performance is compared in [5]. The complexity arises because each performance point can be obtained by CMP and SMT using several combinations of frequency and processor micro-architecture. For a CMP or SMT with a given core architecture, varying the processor frequency provides a continuum of performance points. Any

of these points could be achieved either as a fixed-frequency design or in a system with DVS support. Data for CMP and SMT systems were collected using all combinations of core architectures and frequencies.

For all systems and workloads considered in [5] and for all performance regions, CMP architecture gives the least EPI. It is also found that the best SMT and the best CMP configuration for a given performance target have different architecture and frequency/voltage. Therefore, their relative energy efficiency depends on a subtle interplay between various factors such as capacitance, voltage, IPC, frequency and the level of clock gating, as well as workload features. Although CMP shows a clear energy advantage for four-thread or higher workloads, it comes at the cost of increased Silicon area. Therefore paper investigated a hybrid solution where a CMP is build out of SMT cores, and found it to be an effective compromise.

The work in [6], relates the techniques to increase Instruction level parallelism by improving balanced scheduling with compiler optimization. This study combines Balanced Scheduling with three compiler optimization: Loop Unrolling, Trace Scheduling and Locality Analysis. The researchers "Huiyang Zhou and Thomas M. Conte" developed a method to improve Memory Level Parallelism [7]. This technique parallelizes sequential cache misses speculatively. Value prediction has great potential to enhance MLP by overlapping sequential cache misses.

3. PARALLEL PROGRAMMING METHOD

The main objective is to accelerate the multimedia application, thereby reducing the completion time. The objective is accomplished by making use of the concept of Multi-Grain parallelism (MGP) in the Chip Multiprocessor (CMP). To exploit the different levels of parallelism, p-thread notion is used.

POSIX Thread (p-thread) is a standard for programming with threads and it defines a set of C-types functions and constants. More widely, p-threads are a technique that a program can spawn concurrent units of processing that can then be consigned by the Operating System to multiple processing cores.

Multithreading increases resource utilization by multiplexing the execution of multiple threads on the same pipeline. Clearly, the advantage of multithreading is achieving the high speeds by allocating multiple threads to multiple processing cores, as all cores of CPU or all CPU's if more than one is used operates at the same time.

This method exploits the parallelism from the following levels: Data Level Parallelism, Instruction Level Parallelism, Thread Level Parallelism and Memory Level Parallelism.

3.1 Data Level Parallelism

In parallel computing environment, Data parallelism is a form of parallelization of computing in multiple processors. Data Parallelism focuses on distributing data across different parallel computing nodes. The concept used to achieve Data Level Parallelism is Single Instruction Multiple Data (SIMD) architecture.

In a multiprocessor system executing a single set of instructions (SIMD), data parallelism is achieved when each processor performs the same task on different pieces of distributed data. In some situations, a single execution thread controls operations on all pieces of data. In others, different threads control the operation, but they execute the same code.

In this parallelism the simple ALU functions like addition and multiplication are considered which are stored in a register. The serial execution of the program is done by dividing to tasks and its real time of execution is noted.

The parallel programming is done using POSIX threads where the different tasks (addition & multiplication) are divided into threads and these threads perform tasks simultaneously thereby reducing the real time. Here the main thread holds the control and wait until the other thread execution is completed simultaneously.

A single thread performs the given tasks completely at a time indicating SIMD (Single Instruction Multiple Data).

3.2 Thread Level Parallelism

In thread-Level Parallelism (TLP) instead of having to wait for other threads, it has a capability that enables a program, often a high-end program such as a data or web application to work with multiple threads at the same time. Therefore programs that support this ability are able to do a lot more even under high levels of workloads.

Here we divide the tasks to different threads and one thread is meant to perform a single task and other threads will perform other tasks. The control is taken by the main threads and waits for the rest threads to complete the tasks. The serial version of the execution takes more time comparatively than the parallel which can be observed very clearly.

Hence here one can observe the efficiency of having parallelism concept instead of sequential in multi-core systems using POSIX threads. Comparatively Thread Level Parallelism achieves the highest level of Parallelism amongst other levels of parallelism.

3.3 Instruction Level Parallelism

The potential overlap among instructions is called instruction level parallelism (ILP). It is a measure of how many of the operations in a computer program can be performed simultaneously. Here the superscalar technology

can be adapted for better utilisation of Instruction level parallelism.

It deals with simultaneously performing of all the tasks at a time. The same tasks are performed by threads where there is no inter dependency of data.

Suppose consider,

A=(X*X) +(Y*Y) + (Z*Z)

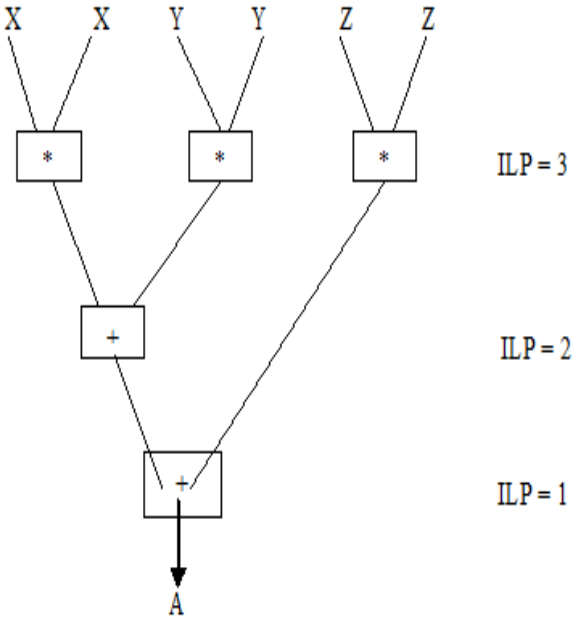

Fig 1: Figure for Instruction Level Parallelism

In parallel executions, different tasks are executed simultaneously. The instruction level parallelism is achieved with real time analysis and comparison with sequential execution is also noted.

3.4 Memory Level Parallelism

Memory Level Parallelism (MLP) is a term in computer architecture. It refers to the ability to have pending multiple memory operations at the same time, in particular cache misses or translation look aside buffer misses. The concept used to achieve Memory Level Parallelism (MLP) is the pre-fetch technique.

There are two ways in which pre-fetching can occur:

- Initiated by hardware,
- Initiated by software.

In this implementation, software pre-fetch is used. Software pre-fetching involves identifying when your application will need a particular set of data, then using special pre-fetch instructions to tell the processor to get this data in advance. The time consumption for both without pre-fetch and with pre-fetch was noted. There was a less time consumption comparatively with using pre-fetch instructions.

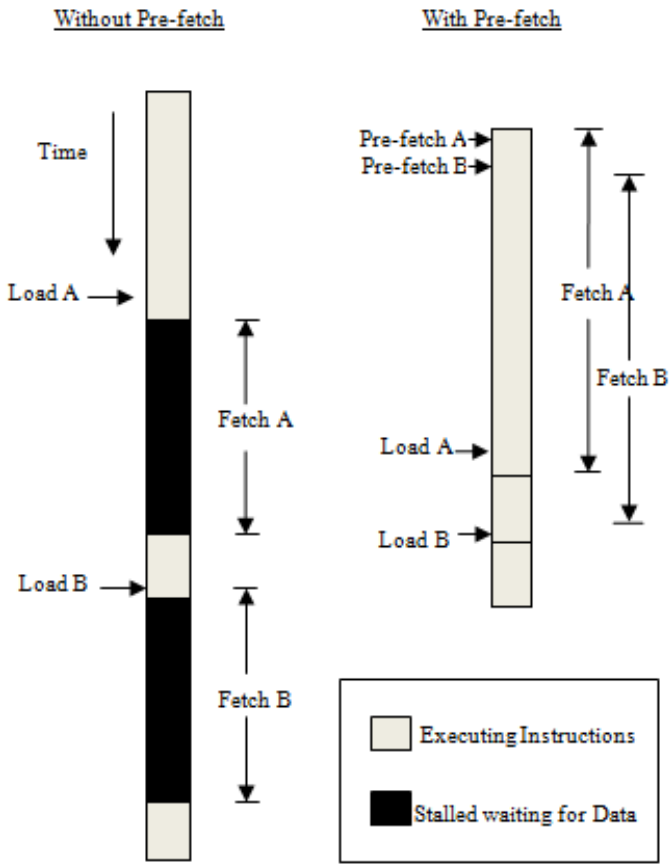

Fig 2: Simple illustration for without pre-fetch and with pre-fetch [internet]

Figure 2, explains about the concept used for Memory Level Parallelism, both for with and without pre-fetch.

4. SIMULATION RESULTS

Table 1: Summary Table

Level of Parallelism	Serial Operation (Seconds)	Parallel Operation (Seconds)
Data Level and Thread Level Parallelism	0.029	0.013
Instruction Level Parallelism	0.013	0.005
Memory Level Parallelism	0.75	0.68

The table 1 shows the summary of the simulation result. It shows the time taken to perform both serial and parallel operation in Data Level Parallelism, Thread Level Parallelism, Instruction Level Parallelism and pre-fetch in Memory Level Parallelism.

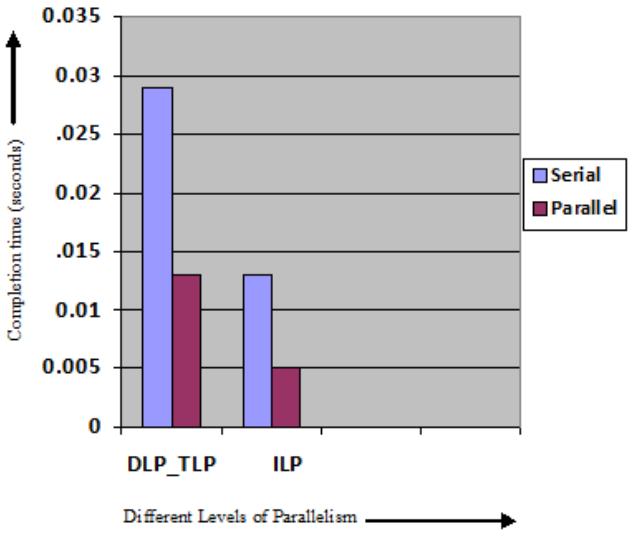

Fig 3: Summary Bar Chart for DLP_TLP and ILP

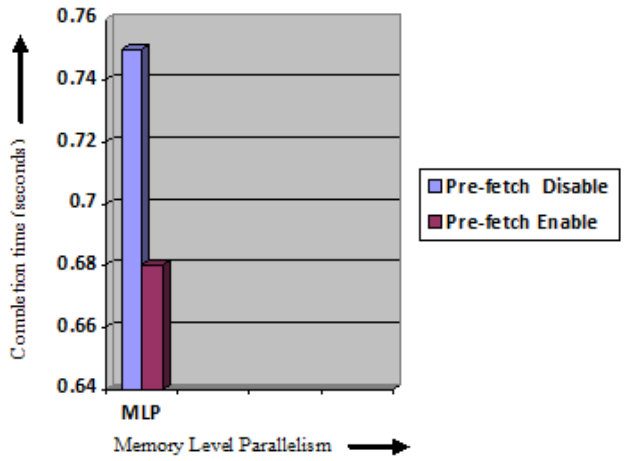

Fig 4: Summary Bar Chart for MLP

Bar Chart shows the time differences for different levels of parallelism.

Figure 3 shows the bar chart for the obtained simulation results for Serial computation and parallel computation for Data Level, Thread Level and Instruction Level. Here the Data and Thread levels are integrated which is shown as DLP_TLP.

Figure 4 shows the bar chart for the simulation result for Memory Level. Memory level is implemented using the pre-fetch concept. It shows the time differences for with pre-fetch and without pre-fetch operation. From the result, time taken with pre-fetch is more as compared with without pre-fetch.

5. CONCLUSIONS AND FUTURE SCOPE

In this implementation, different methods are implemented to exploit different levels of parallelism. POSIX Thread concept is used to implement the Data Level, Thread Level and Instruction Level Parallelism. Instruction Level Parallelism is based on superscalar architecture. Memory level is achieved using the Pre-fetch concept. The simulation results show that all the levels of parallelism can be achieved.

The time difference for serial as well as parallel computation is also compared. From the simulation result it is shown that completion time is more for serial computation as compared to that of parallel. Hence parallel computation can accelerate multimedia application in Chip Multiprocessors.

As a future research, the following work can be performed; integration of all the levels can be performed along with extension to multiple processors on chip. It can be further configured for different workloads especially in case of multimedia application.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my guide Mrs. Kavitha. V, Ph D Scholar, Jain University for her timely guidance and encouragement to make this project, a success

REFERENCES

[1]. Xiaoping Huang, Xiaoya Fan, Shengbing Zhang And Liwen Shi, “Investigation On Multi-Grain Parallelism In Chip Multiprocessor For Multimedia Application”, proceeding of IEEE 2009, Computer School, Northwestern Polytechnical University, China.
[2]. Man-Lap Li, Ruchira Scscnka, Sarita V. Adve, Yen-Kuang Chen, Erid Debes, “The ALPbench Benchmark Suite For Complex Multimedia Application”, Proceeding of the 2005 International Symposium On Workload Characterization, October, 2005.
[3]. KunleOlukotun, LanceHammond of Stanford University and James Laudon of Sun Microsystems, “Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency”, 2007.
[4]. Bushra Ahsan, Fatma Omara and Mohamed Zahran,Department of Computer Science, “Chip Multiprocessor: Challenges and Opportunities”, INFOS2008, March 27-29, 2008 Cairo-Egypt.
[5]. Ruchira Sasanka, Sarita V. Adve, Yen-Kuang and Eric Debes , “The Energy Efficiency of CMP v/s SMT for multimedia workloads”, June 26–July 1, 2004, SaintMalo, France, Copyright 2004.
[6]. Jack L.Lo and SusanJ.Eggers; “Improving Balanced Scheduling with Compiler Optimization that Increase Instruction Level Parallelism”, Department Of Computer Science and Engineering, University Of Washington, 1995.
[7]. Huiyang Zhou and Thomas M. Conte, Department of Electrical and Computer Engineering,“Enhancing Memory Level Parallelism via Recovery-Free Value Prediction”, June 23-26, 2003, SanFrancisco, California, USA, Copyright 2003.
[8]. Kostas Bousias, Nabil Hasasneh, Chris Jesshope, “Instruction-level parallelism through Micro threading- A Scalable Approach to Chip Multiprocessor”, Computer Journal, March 2006,49(21): 211-233.
[9] http://en.wikipedia.org/wiki/C_(programming_language)

[10].http://en.wikipedia.org/wiki/C++_(programming_langu
age)
[11]. http://en.wikipedia.org/wiki/ubuntu
[12]. http://en.wikipedia.org