Cache memory a brief study

Vishal Bhalla;
B.tech (Computer Science & Engineering
Dronacharya college of engineering.
Gurgaon (Haryana), INDIA
vishalbhalla.08@gmail.com

Virender Gupta
B.tech (Computer Science & Engineering
Dronacharya college of engineering.
delhi, INDIA
virendergupta93@gmail.com

Ankit Gahlot
B.tech (Computer Science & Engineering
Dronacharya college of engineering.
delhi, INDIA
ankitgahlot51@gmail.com

ABSTRACT

Advancements in multi-core have created interest among many research groups in finding out ways to harness the true power of processor cores. Recent research suggests that on-board component such as cache memory plays a crucial role in deciding the performance of multi-core systems. In this paper, performance of cache memory is evaluated through the parameters such as cache access time, miss rate and miss penalty. The influence of cache parameters over execution time is also discussed. Results obtained from simulated studies of multi-core environments with different instruction set architectures (ISA) like ALPHA and X86 are produced.

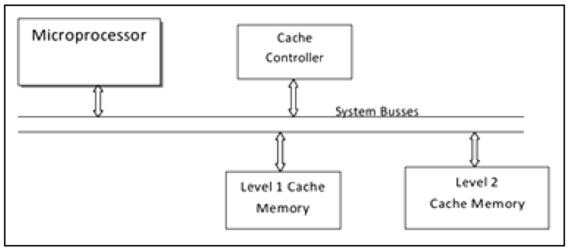
INTRODUCTION

One of the important factors that influence execution time of a program is the cache access time . Cache memory provides a quicker supply of data for execution by forming a bridge between the faster processor unit on one side and the relatively slower memory unit on the other side While it is well known that cache memory helps in faster access of data, there is considerable research groups to understand the impact of cache performance on the execution time for obtaining better performance of multi-core platforms . Latest advancements in cache memory subsystems for multicore include increase in the number of levels of cache as well as increase in cache size. Traditional uni-core systems had a dedicated caching model whereas the recent multicore systems have a shared cache model. As a result of sharing and increase in the number of levels of cache, the cache access time increases and tends to consume a higher percentage of memory access time which in turn affects the execution time. A related issue which assumes significance is the effect of instruction set architectures on cache performance wherein different instruction set architectures influence the cache memory access time of programs.

In this paper we have tried to estimate the access time of shared cache memory on multi-core platforms for different ISA's. Results pertaining to execution time are also presented.

SYSTEM ARCHITECTURE OVERVIEW

Cache controller that communicates between microprocessor and cache memory to carry out memory related operations. The functionality of the design is explained below.



Cache controller receive address that microprocessor wants to access Cache controller looks for the address in L1 cache. If address present in L1 cache the data from location is provided to microprocessor via data bus. If the address not found in L1 cache then cache miss will occur. Cache controller then looks same address in cache L2. If address present in L2 cache the data from location is provided to microprocessor. The same data will replace in cache L1. If the address not found in L2 cache then cache miss will occur. In our paper work we design single cache memory for detecting miss rate in cache memory and less power consumption. This work will be used in design of FPGA based 2-way, 4-way, 8-wayset associative cache memory and used in cache controller for tracking induced miss rate.

Features of Cache

A cache has many features which made the modern computing systems possible.
it has a features such as:
*Capacity
It defines how much of the data can be stored into the cache.
If the capacity is 32KB, then it can only store 32KB data, But Actually, such a cache requires more than 32KB of data because some memory would be taken by tag array that is cache directory.
*Line length
Line length is basically the size of cache's block. If a cache with 32byte cache line has a miss, it brings a 32byte block of data containing address of the miss into cache.
The alignment of the cache lines is in such a manner that:-
*the lower address bits determine which blocks bytes if cache contains the data.
*the higher address bits determine if there is a hit/miss.

Basic Instruction
Processor references that are found in the cache called as cache hit else it is called a cache miss. On a cache miss the control mechanism references the data from the memory and places it in the cache.
The whole of the interaction of the cache and the processor is based on the hit or miss logic.

Related work

Substantial work is yet to be carried out on performance of cache memory systems for multicore
platforms. Some of the previous work have been considered wherein:
1. In by S. Laha et. al., Accurate low-cost methods for performance evaluation of cache
memory systems have been discussed. In this work, results pertaining to trace driven simulation of predicting mean miss rate of cache have been presented.
2. Similarly, in Cache performance of SPEC92 benchmark suite has been discussed in
detail by J.E.Gee et.al. In this work issues pertaining to instruction cache miss ratio and data cache miss ratio have been discussed.
3. Our previous works have been on understanding the performance of multi-core platforms
[6] and on the performance of cache on machines with X86 ISA . Issues related to cache memory access and execution time of programs have been discussed using simulation tools such as DINERO IV and CACTI.
4. The work by A.C. Schneider et. al., on dynamic optimization techniques deals with

both ALPHA and X86 architectures on multi-core platforms.
We use similar techniques described in the above said work to compare the performance of
ALPHA and X86 ISA.

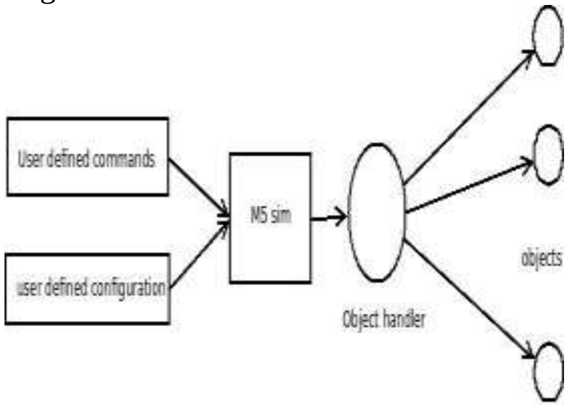
Problem statement
The primary objective of this paper is the evaluation of the impact; caches have on different
instructions set architectures. This is achieved by taking one particular benchmark from
SPLASH2 and finding the access time on ALPHA ISA using M5sim and comparing the results
with the results obtaining using CACTI on X86 ISA. The other issue that is addressed in this
paper is the measure of execution time for varying sizes of cache. Also the impact of cache on
execution time is demonstrated through simulation results using SPLASH-2 benchmark suite on
M5sim. Here we have tried to find out whether benchmarks running on different instruction set
architectures with a specific underlying hardware configuration produce results that are
qualitatively similar.
The access time plays a key role in determining the execution time of any process. In a multi-core
environment as the number of levels of cache increases, the cache access time tends to consume a
major percentage of memory latency. Since the state of art multi-core systems have almost three
levels of caches, it becomes essential to understand the impact of hierarchical cache design. This
motivated us to study the influence of cache on both access time and execution time.

Section 2 describes the various tools used in the experiments followed by Section 3 describing
the experimental setup and the parameter set used in the experiments. Section 4 discusses the
summary of results followed by Section 5 dealing with future research and conclusion.

TOOLS USED

M5SIM

M5 is an emulation tool that is capable of performing event driven simulation. It enables users
to simulate a multi-core environment with error modularity close to hardware. The represented
model of system can be viewed as a collection of objects like CPU cores, caches, input and output
devices. M5sim uses Object, Event, Mode driven architecture. Fig. shows the basic block
diagram of M5sim.

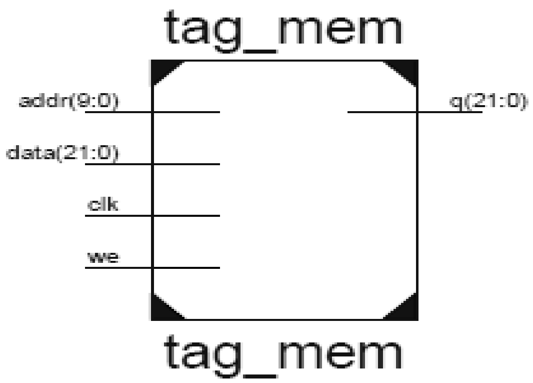


Objects are components such as CPU, memory, caches which can be accessed through interfaces.
Events involve clock ticks that serve as interrupts for performing specific functionality. There are
two basic modes of operation namely: full system mode and system call emulation mode. The
major difference between the two modes of operation is that, the later executes system calls on a host machine whereas the former

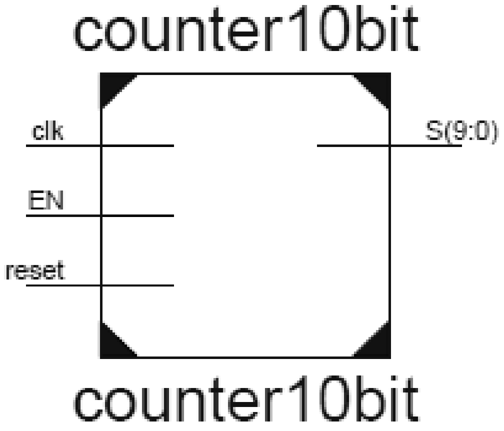
emulates a virtual hardware whose execution is close to the real
system. In this paper, all the experiments are conducted using full system mode for alpha instruction set architecture.

DESIGNED WORK

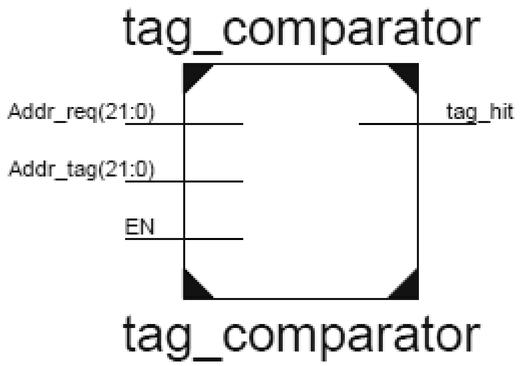
In our designed work, we designed cache memory L1 using FPGA. In future with reference to designed work we can design set associative cache memory as set associative cache strike a good balance between lower miss rate and higher cost. Cache memory consist Cache tag memory, cache data memory, cache tag comparator and counter. Our paper work deal with detection of cache misses. We designed cache tag memory, cache tag comparator and 10-bit counter and need not to design cache data memory for detecting cache misses as address is stored in tag memory and data is stored data memory. Address Requested by microprocessor is compared with address of data in main memory those are stored in tag memory. The cache tag memory stored address of
shows the diagram of 10-bit counter. For each clock pulse counter is incremented by one. 10- bit counter provides 10-bit addresses to cache tag memory during writing and reading operation. The 22-bit cache tag comparator Cache tag comparator is used to compare address requested by microprocessor and address stored in tag memory. 22-bit cache tag comparator is modelled using behavioural style of modelling in VHDL. data in main memory. Cache tag memory is modelled using behavioural style of modelling in VHDL (for example: 22-bit cache tag memory).



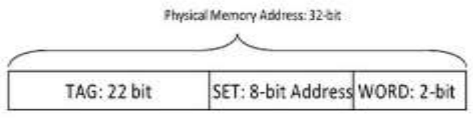
shows the diagram of cache tag memory of 22 bit. When „we‟=1 it performs writing operation and „we‟=0 it performs reading operation. Data stored in the tag memory will be a addresses of main memory. The 10-bit counter used to generate 10 bit address for writing and reading operation of cache tag memory**.** 10-bit counter is modelled using behavioural style of modelling in VHDL.



shows the diagram of 10-bit counter. For each clock pulse counter is incremented by one. 10- bit counter provides 10-bit addresses to cache tag memory during writing and reading operation. The 22-bit cache tag comparator Cache tag comparator is used to compare address requested by microprocessor and address stored in tag memory. 22-bit cache tag comparator is modelled using behavioural style of modelling in VHDL.



shows the diagram of 22-bit cache tag comparator .22-bit Cache tag comparator is compare 22-bit address requested by microprocessor and 22-bit address stored in tag memory. Address requested by microprocessor is 32-bit out which 22-bit address is used to compare with tag address.8-bit address is used as a set address and 2-bit address is used as a offset word address. To design set associative cache memory, we have to make number of sets of such cache memory which will be our future work



The implementation of cache memory requires cache tag memory, 10-bit counter and cache tag comparator as shown in figure

REFERENCES

[1] Jongsok Choi, Kevin Nam, Andrew Canis, Jason Anderson, Stephen Brown, and Tomasz Czajkowski, "Impact of Cache Architecture and Interface on Performance and Area of FPGA-Based Processor/Parallel-Accelerator Systems" ECE Department, University of Toronto, Toronto, ON, Canada ,Altera Toronto Technology Centre, Toronto, ON, Canada

[2] Maciej kurek, ioannis ilkos, wayne luk, "Customizable security-aware cache for fpga-based soft processors", Department of computing, imperial college London

[3] Nawaf Almoosa, Yorai Wardi, and Sudhakar Yalamanchili, Controller Design for Tracking Induced Miss-Rates in Cache Memories, 2010 8th IEEE International Conference on Control and Automation Xiamen, China, June 9-11, 2010

[4] Computer architecture and organisation : John .p. Hayes(Mc Graw hill publication)

[5] Computer architecture: Forth Edition, John L. hennessy and David A. Patterson.

[6] Vipin S. Bhure , Praveen R. Chakole, **"**Design of Cache Controller for Multi-core Processor System" international Journal of Electronics and Computer Science Engineering, ISSN: 2277-1956