A LITERATURE SURVEY ON CACHE MEMORY

Y POORNA CHANDRA
2nd Year, ECE,SSE
SIMATS

S AFREED
2nd Year, ECE,SSE
SIMATS

ABSTRACT

Nowadays, processing speed is one of the most important performance criteria of modern multicore processors. For achieving a higher processing speed of processor various components are used, in which cache is one of them. Cache Memory is extremely quick memory placed between central processor & Main Memory. Cache memory plays an important role when deciding the performance of multi-core systems. Processor speed is increasing at a very fast rate so a very high speed cache memory is needed for matching the speed of the processor. Cache systems are on-chip Cache memory part accustomed store knowledge. Cache is a buffer between a central processor and its main memory. Cache memory is employed to synchronize the info transfer rate between central processor and main memory. As cache memory nearer to the microchip, it is faster than the RAM and main memory. The advantage of storing knowledge in cache, as compared to RAM, is that it has faster retrieval times, but it has the disadvantage of on-chip energy consumption. In this paper, performance of cache memory is evaluated through following factors cache time interval, miss rate and miss penalty.

Keywords: *Cache memory, access, hit ratio, addresses, mapping, on-chip memory*

INTRODUCTION:

Dictionary that means of cache is "A assortment of item of an equivalent sort hold on in a very hidden or inaccessible place". Caches are generally the top level of the memory hierarchy and are made of SRAM (Static Random access Memory). The main structural difference between a cache and other level in the memory hierarchy is that caches use hardware to locate memory addresses whereas other memories use software or a combination of software and hardware. Today caches have become an integral part of all processors. Performance improvement of microprocessors historically comes from both increasing the speed or frequency at which the processors run and by increasing the amount of task performed in each cycle. The increasing number of transistors on a chip has led to different ways of increasing parallelism. In the close to future, microprocessors are ready to execute multiple processes/threads at the same time and exploit process/thread-level similarity.

Cache memory was first seen in the IBM system/360 Model 85 in 1960. In 1980s, the Intel 486DX microprocessor introduced an on chip 8 KB L1 cache for the first time. Cache memories are small fast memories used to temporarily hold the contents of portions of main memory that are likely to be used. In Cache memory often used information or directions are unbroken in order that it is accessed at a awfully quick rate up overall performance of the pc. Commonly during a processor 1st level cache (L1) reside within the processor and second level cache (L2) and third level cache (L3) ar on separate chip.

Cache memory is divided into two different parts; one is cached data memory and another is cache tag memory. Cache data memory contains various collections of memory words called cache block or line or page. Each cache block has a block address or tag. Collection of all block addresses or tags is called cache tag memory. Whenever central processor needs any knowledge or instruction it sends address to the cache memory, address is searched in cache memory firstly, if data is found in cache it is given to the CPU. Finding a data in cache memory is named as cache HIT. If data isn't found in cache then it's referred to as cache MISS, in this case address is searched in main memory for the data or instruction. After data is received from main memory, a block of data is transferred from main memory to cache memory so all any request is consummated from cache. Performance of cache memory is measured in HIT Ratio.

Hit Ratio denoted by H is defined as the ratio of the total number of hits and total no. of hits and misses.

Hit ratio= Total number of hits/(Total no.of hits+ Total no.of misses)

Miss Ratio is denoted by M is defined as

M = 1 – Hit ratio

For measuring the performance of multi core processor we use Instruction per Cycle (IPC) and Cycle Per Instruction (CPI). IPC is number of instructions are executed in one cycle. IPC= 1/(Number of Instruction)

CPI is number of cycles are needed to execute one instruction.

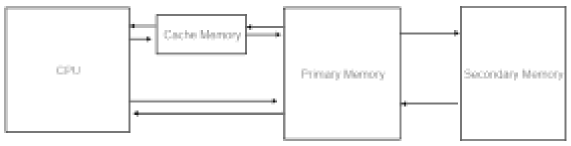

Fig 1:-Block diagram of cache memory

Cache memory is based on principal of section of References.

By this we tend to mean once a program executes on a pc, connected storage locations being often accessed. Temporal section refers to the utilize of specific data and/or resources among comparatively little time durations. Spatial locality refers to the employment of data parts among comparatively shut storage locations. The transformation of data from main memory to cache memory is referred as a mapping method. Basically, there are three methods for mapping addresses to cache locations - direct mapping, associative mapping and set associative mapping. Direct mapping is the simplest technique which maps each block of main memory into only one possible cache line. In associative mapping each block of main memory maps into any line of the cache. In set associative mapping cache is divided into sets, each of which consists of cache lines or blocks and each block of main memory maps into any of lines of set.

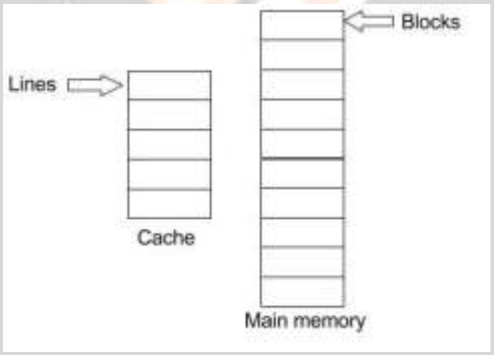

Fig 2:- Cache mapping

Several of the cache design parameters will have a significant effect on system performance. The choice of line size is important. Selecting the degree of set associativity is another necessary trade-off in cache design. However, increasing the degree of set associativity will increase the value and quality of a cache, since the number of address comparators needed is equal to the degree of set associativity. Recent cache memory analysis has created the fascinating result that an on the spot mapped cache can typically outperform a group associative (or totally associative) cache of identical size even though the direct mapped cache will have a higher miss ratio. This is because the increased complexity of set associative caches significantly increases the access time for a cache hit. As cache sizes become larger, a reduced access time for hits becomes more important than the small reduction in miss ratio that is achieved through associativity.

LITERATURE REVIEW ON CACHE MEMORY:

Our Review is based on various cache memories and performance issues in multicore processors.

To obtain greater parallelism the cache can be loaded in a bit slice mode so that it contains a slice of two or more bits from each word instead of single bit from each word. Majority of memory requests are executed in cache. Microelectronics memories can be embedded effectively as cache memories. Instructs the computer system how to load the cache. The systems can make effective use of cache without any explicit guidance from programs. Cache memory has limited capacity. Not suitable for high performances computer and Cached data is stored only so long as power is provided to cache [1]. Cache memory serves the buffering function of reducing traffic between processor and main memory very substantially. Problems of multiple copies of data must be resolved by using cache memory. It's either read or a write miss - some limited data suggests that this results in slightly better performance and it reduce memory contention. Reuse and locality of programs in computer. Cache memory is not flexible. In a direct mapped cache, possible to assume a hit and continue. Recover later of miss. Miss ratio is high in cache memory [2]. Analytical determination of the optimum capacity of a cache memory with given access time and of its matching storage hierarchy has been achieved based on an earlier model of linear storage hierarchies. The resultant formulas show explicit functional dependence of the cache's capacity, and the capacities and the access times of other storage levels, upon the miss ratio parameters and technology cost parameters as well as the required hierarchy capacity and the total allowable cost. In particular, the cache size is directly proportional to the hierarchy cost, and larger cache access time or larger miss ratio requires a larger cache for optimum performance. Cache memory requires an inter process call for each cache access and Difficult to implement. Use larger cache size with additional block frames to avoid contention [3]. Cache memory gives a good indication of relative performances and it predicts page fault rates reasonably well. It is analytically tractable. Replacement algorithms of block when new block read into cache (Associative search). No page replacement policy. Cache hit performance is poor and Consuming different amounts of power in their on and off states and Access time is high [4]. The very fast Josephson NDRO cache memory design are structured and are interfaced with one another. The discussion has centered on the performance of a IK-bit array, four of which we expect ultimately to place onto a 6.35- x 6.35mm chip. Extremely fast switching (<10ps). Extremely low power dissipation (500 nW per circuit). Operation at very low temperatures (=4k). Josephson devices are attractive for ultrahigh-performance computers. Conducting strip lines and ground planes with zero electrical resistance. Bus traffic memory is high and Memory bandwidth is high. Complex circuitry required to examine tags of all caches line in parallel [5].

The use of cache memory, however, has often aggravated the bandwidth problem rather than reduce it. Reducing processor memory bandwidth. On chip memory perform as fast as the more powerful processor and Maximizing the hit ratio. Its Minimizing the access time to data in the cache. Minimizing the overheads of updating main memory, maintaining multi-cache consistency [6]. For single-chip computers to achieve high performance, on-chip memory must be allocated carefully. Pin bandwidth is limited. It consistently works well. It takes dynamic program behaviour into account. It outperforms a data cache by nearly a factor of two in both speed and cost. It potentially can use better algorithms with information available at compile time to preload data into registers and purge data more effectively than a data cache. Registers should perhaps be thought of as local memory. Requires large number of memory cycles. The register should be updated in every cache area. Time requirement is high for transferring the data between memory and cache [7]. A cache memory reduces the main memory traffic for each processor. Require less time to transmit between main memory and cache. Require fewer memory cycles to access if the main memory breadth is slim. Require fewer address tag bits in the cache. Minimize bus cycle time and Increase bus width to Improve bus protocol. Frequency based replacement schemes. Cache memory size is high. Cache memories are most likely to impact system performance [8]. Reducing Conflict Misses: Miss Caching and Victim Caching. Combining Long Lines and Stream Buffers. Stream buffers that may prefetch down many streams at the same time. Reduce cache miss rates and improve system performance. Effective increase in data cache size provided with stream buffers and victim caches using 16B lines. Can effect overall system performance and Practical only for small caches. Size of data cache is high [9]. Increasing Write-Back Cache Bandwidth. Reducing Write-Through Cache Traffic. Limiting code exploitation these directions to the machines with cache line sizes capable or smaller than that assumed within the allot directions. Extra execution overhead for the cache allocation directions. Cache Memory affects the execution time of a program and high hardware complexity. It requires more cycles are required [10].

High hardware complexity. It is not used in hard real-time systems. No conflict when accessing instruction. Scalability issues [11]. Cache memory being the common technique to bridge the speed gap. Excess CPI with cache. Replacement policy (random, LRU, FIFO) Write policy (CBWA, WT).Full associative (Area = PLA + Data + Tag.). Memory must be updated with each cache. Could run into issues syncing caches. It cannot stores the previous state. Pre- fetch is not done [12].It can be used in hard real-time systems. Reducing the upper limit execution time. It is restored to its previous state when the preempted task is reloaded. Worst case specifications, frequently resulting in under-utilization of the processor. Extremely large caches (up to 511GB). Affecting the architecture of the next level of the memory hierarchy and performance is low in real time systems. Worst case execution time is high [13].Selective victim caching, for improving the miss-rate of direct-mapped caches. It does not affect the cache access time. Places data in the main cache. Improvement in miss ratio. Improving number of interchanges between the main direct-mapped cache and the victim cache over simple victim caching. High access time. No algorithm when applied to data caches. Requires more random patterns of data references [14]. Cache memories are small fast memories used to temporarily hold the contents of memory. Caches became an integral a part of all processors. Virtually addressed caches do not require address translation during cache access .Multiple-access cache, Decoupled cache. More and more silicon area is being dedicated to the on-chip caches .Processor become increasingly superscalar. Requires more aggressive designs to handle multiple requests per cycle [15].

Decide what to try and do – execute directions to calculate new positions etc. Actuate – control the process by giving a signal to a controlling device (motor, relay etc.)Inline probes are placed directly in the application code. Bottleneck problem for an embedded system. High peak power and memory is unable to keep pace with the CPU [16]. Executing multiple processes at the same time like in CMP/SMT systems. Reducing cache interference. Dynamically partitions the cache amongst the processes. Processor with multiple functional units. While executing processes require large caches. A small number of additional counters are required. Cannot improve the performance if caches are too small for the workloads [17].3D graphics cache system to increase memory utilization. Solving a memory bottleneck problem for an embedded system. Decrease an intensive memory access during a short period and maximize memory utilization. Reduce power consumption by supporting an AXI low power interface. Improving on-chip bus. Implementation is difficult [18]. High speed multicore processor. To improve the timing. By aligning cell instances into "vectors", we reduce control and clock loading. Largest Benefit Quick feedback to the DE for Area & Timing. Quick to proto-type different data path configuration. Replacement circuit becomes more complicate. High complexity. Design of cache memory is complicated [19]. Design of cache memory on FPGA for police investigation cache miss. Higher performance with lower power.

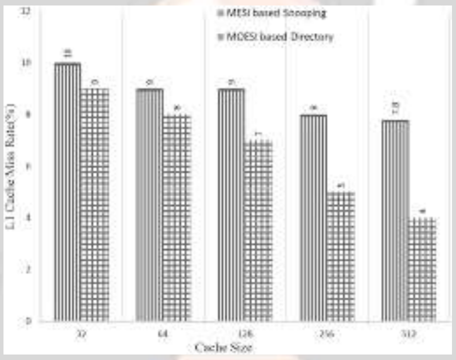

Fig-3:- Cache size

Cache controller for chase the induced miss rate in cache memory. On-chip energy consumption. No replacement policies. Time execution is high [20].

Faster replacement ways generally keep track of less usage info. To reduce the amount of time needed to update that info. In the case of direct-mapped cache, no information. Video and audio streaming applications. Cache information that will be used again (cache pollution). Number of transistors on chip increasing on the far side one billion. World scientific codes already exceed the maximum capacity. Scientific computing typically require several Megabytes up to hundreds of Gigabytes of memory [21]. The rate of conflict misses is reduced by exploitation larger block size, larger cache and way prediction methods. However, exploitation larger block size could increase miss penalty, reduced hit time and power consumption. On the opposite aspect, larger cache produces slow access time and high cost. It is associated with cache coherence problem. Higher associativity turn out quick interval however they need low cycle time. Problem Matrix multiplication of different size. Hard to identify a certain cache optimization technique. Larger cache produces slow access time and high cost [22]. Higher associatively turn out quick interval however they need low cycle time. Victim cache reduces miss rate at a high cost comparing to Cache miss look aside. In all we will say that there's a space for up performance of cache memory to a really giant extent. Cache Memory affects the execution time of a program [23]. Flash memory provides high performance, high capacity, and stable quality of service (QOS). Increases the amount of data processed by the processor. Cache size increased from 2KB to 16KB and performance increase was 273.8% for random write, 214.4% for random read. Traffic increases between processor and cache memory. More cache misses. Cache misses cause cache miss penalty to degrade overall storage performance [24]. The performance of cache on the basis of replacement policies such as LRU, FIFO and Random. They found that the LRU policy in the data cache has better performance than FIFO and Random. The instruction cache replacement policies. They proposed a new loop model. In its loop model, they found that random replacement has performed better than LRU and FIFO. However, each simulation has different cache sizes, different cache associativity and different benchmarks. Therefore, the performance comparisons of policies are less accurate. A unified simulation should be implemented for all policies to compare their performance [25].

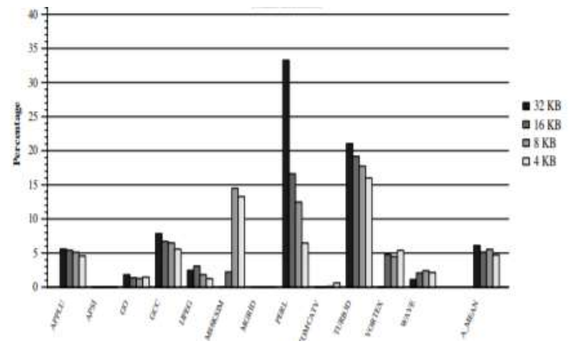

Fig 4:- Traffic reduction

CONCLUSION:

With all the references of this papers, we discussed cache memory and how to improve the performance of cache memory. So we can say that Cache memory may be a high-speed random access memory that is employed by a system Central process unit for storing the data/instruction temporarily. It decreases the execution time by storing the foremost frequent and most probable data and instructions "closer" to the processor, wherever the systems central processing unit will quickly catch on. Future caches are larger and smarter, the data structures presently used in real. Today's applications in scientific computing usually need many Megabytes up to many Gigabytes of memory.

REFERENCES:

1. H. A. Curtis, "Systematic procedures for realizing synchronous sequential machines using flip-flop memory: Part I," IEEE Trans Computers, vol. C-18, pp. 1121-1127, December 1969.
2. R. 0. Winder, "A Data Base for Computer Performance Evaluation", presented at IEEE Workshop on System Performance Evaluation, Argonne, Ill., October 1971, and published in COMPUTER, vol. 6, no. 3, March 1973.
3. C. Chow, "An optimization of memory hierarchies by geometi programming," in Proc. 7th Annu.. Sa. Syst., 1973, p. 15.
4. AVEN, O I, ET AL Some results on distribution-free analysis of pagmg algorithms IEEE Trans Comptrs C-25, 7 (July 1976), 737-745.
5. P. Gueret, A. Moser, and P. Wolf, IBM J. Res Develop. 24 (1980, this issue).
6. [Amdahl 82] C. Amdahl, private commuvtict~tiovt, March 82.
7. Hasegawa, M. and Y. Shigei, "High-Speed Top-Of-Stack Scheme for VLSI Processors: a Management Algorithm and Its Analysis," Proceeding of the 12th Annual International Sym- posium on Computer Architecture, June, 1985.
8. DONALD C. WINSOR AND TREVOR N. MUDGE. “Analysis of Bus Hierarchies for Multiprocessors”. The 15th Annual International Symposium on Computer Architecture Conference Proceedings, Honolulu, Hawaii, IEEE Computer Society Press, May 30–June 2, 1988, pages 100–107.
9. Baer, Jean-Loup, and Wang, Wenn-Hann. On the Inclusion Properties for Multi-Level Cache Hierarchies. In The 15th Annual Symposium on Computer Architecture, pages 73-80. IEEE Computer Society Press, June, 1988.
10. Smith, Alan J. Second Bibliography on Cache Memories. Computer Architecture News 19(4):154-182, June, 1991.
11. H. B. Bakoglu, G. F. Grohoski, and R. K. Montoye, “The IBM RISC system16000 processor: Hardware overview,” IBM J. Res. Develop., vol. 34, no. 1, pp. 12-22, Jan. 19.
12. M.J.Flynn, "Computer Architecture: Concurrent and Parallel Processor Design", Jones and Bartlett, Boston, 1994.
13. Lebeck, A.R. and Wood, D.A. (1994). Cache Profiling and the SPEC Benchmarks: A Case Study. IEEE Computer 27(10), 15-26.

14. D. Stiliadis and A. Varma, “Selective Victim Caching: A Method to Improve the Performance of Direct-Mapped Caches,” Tech. Rep. UCSC-CRL-93-41, U.C. Santa Cruz, 1993 (http://www.cse.ucsc.edu/research/hsnlab/publications.html).
15. J. Peir, Y. Lee, W. Hsu, \Capturing Dynamic Memory Reference Behavior with Adaptive Cache Topology," 8th ASPLOS, Oct. 1998, pp. 240{250.
16. Filip Sebek and Jan Gustafsson. Determining the worst-case instruction cache miss-ratio (paper c). In Proceedings of ESCODES 2002, San José, CA, USA, September 2002.
17. S. Yang, M. D. Powell, B. Falsafi, and T. N. Vijaykumar. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In The 8th High-Performance Computer Architecture, 2002.
18. Kil-Whan Lee, Woo-Chan Park, Il-San Kim, and Tack-Don Han, "A Pixel Cache Architecture with Selective Placement Scheme based on Z-test Result," Microprocessors and Microsystems, Vol. 29, Issue. 1, pp. 41-46, Feb. 2005.
19. O. Temam, "An Algorithm for Optimally Exploiting Spatial and Temporal Locality in Upper Memory Levels," IEEE Trans. Computers, 48, No. 2, pp150-158.
20. 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.
21. N. Ahmed, N. Mateev, and K. Pingali. Tiling Imperfectly{Nested Loop Nests. In Proc. of the ACM/IEEE Supercomputing Conference, Dallas, Texas, USA, 2000.
22. Y. Xu, Y. Li, T. Lin, Z. Wang, W. Niu, H. Tang, and S. Ci, “A novel cache size optimization scheme based on manifold learning in Content Centric Networking,” J. Netw. Comput. Appl., vol. 37, pp. 273–281, Jan. 2014.
23. M. Kampe, P. Stenstrom, and M. Dubois, “Self-correcting LRU replacement policies,” Proc. first Conf. Comput. Front. Comput. Front. - CF’04, p. 181, 2004. [25] M. Kharbutli and Y. S. Y. Solihin, “Counter-Based Cache Replacement and Bypassing Algorithms,” IEEE Trans. Comput., vol. 57, no. 4, 2008.
24. Xilinx, "UG585 Zynq-7000 All Programmable SoC Technical Reference Manual (v1.11)," September, 2016.
25. Doug Burger, Todd M. Austin, “The SimpleScalar Tool Set, Version 2.0”, http://www.simplescalar.com, retrieved as on April 2017.