

SURVEY ON VARIOUS ADVANCED TECHNIQUE FOR CACHE OPTIMIZATION METHODS FOR RISC BASED SYSTEM ARCHITECTURE

¹G.Arun Francis, ²S.Manikandan, ³P.Karthigaikumar,

^{1,2}Assistant Professor, Karpagam College of Engineering, Coimbatore

³Professor, Karpagam College of Engineering, Coimbatore

¹ja.arunji@gmail.com, ²mailingtomani@gamil.com, ³p.karthigaikumar@gmail.com

Abstract: Survey on non-uniform cache design concludes the paper that may be a future approach to an outsized range of core processors. Cache may be a memory in between the processor and also the main memory. A smaller temporary memory that manages the main memory locations and access time thereby will increase speed throughout execution time. With the trend of transient error rate, its turning into necessary to forestall transient errors and supply a correction mechanism for hardware circuits, specially for SRAM cache recollections. Caches measure the most important structures in current microprocessors and, hence, square it measures most prone to the transient errors. This paper at first exploits the same tag bits to enhance error protection capability of the tag bits within the caches. Once information measure fetched from the most memory, it's checked if adjacent cache lines have identical tag bits as those of the info fetched and this paper incorporates a thorough discussion concerning cache and varied mapping technique. Then we have a tendency to shift our focus on cache optimizations and discuss the motivation for doing this on the same, followed by the completely different improvement techniques. any to avoid varied classes of cache misses we have a tendency to discuss completely different sorts of cache technique to reach higher performance. Lastly we have a tendency to discuss few open and difficult problems featured in varied cache improvement techniques.

Keyword: SRAM, Cache, microprocessors, mapping technique.

1. Introduction

The cache memory may be a terribly little however quick quite memory that is associate negotiate between the most memory and also the processor in an endeavor to cut back the range of times the main memory is accessed by programs. The reduction of the common time of main memory access is achieved by

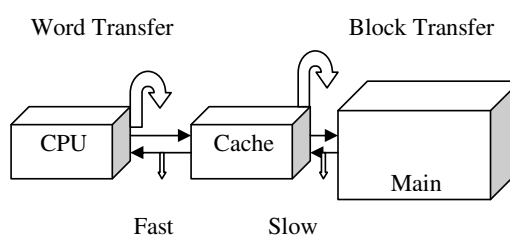


Figure 1. Working of Cache.

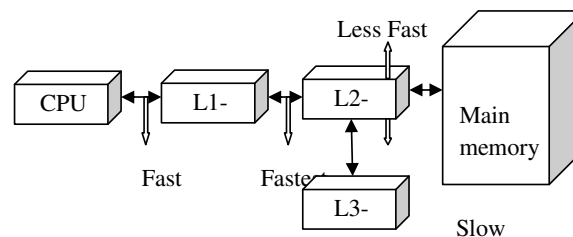


Figure 2. Three Level Cache organization

the cache through storing copies of the information from the most oft used locations of the most memory. parenthetically, once the necessity arises for the processor to scan from or write to a main memory location, the processor first performs a check to ascertain whether or not there's a duplicate of that information within the cache. once the processor finds a duplicate of the info in the cache, it reads from or writes to the cache forthwith which is able to end up to be quicker (for the processor) than reading from or writing to the main memory directly. exploitation the cache memory during this manner aids within the reduction of the common latency of memory accesses. For understanding the structure and operating construct of the cache, the Intel Pentium four processor will be thought-about as associate example. As shown within the Figure one and a couple of the processor (Intel Pentium 4) use 2 cache levels (L1 and L2). L1 is cited because the Level

lcache that is 8KB and 16KB whereas the L2 is referred to because the Level a pair of cache that incorporates a size of 1MB. With the advancement in technology, the sizes square measure unendingly pushed up creating it not uncommon to search out associate L2 size of 4MB. In an endeavor to separate the information As shown in the Figure 1 and 2 ,from the directions in the L1 cache, it would be found that the existence of 2 L1 caches per processor/core is additionally quite common. However, the L2's arrangement is in a very manner that each information and the directions square measure along. The make-up of the L2 is renowned as unified. fashionable desktop associated servers principally have at least 3 freelance caches: an instruction cache, a information cache and a translation-look-aside-buffer (TLB). We have a tendency to evaluate our theme with a antecedently projected competitive theme, in-cache replication (ICR). ICR was originally projected to cut back vulnerability of the info array however also can be applied to cut back tag bits vulnerability. We have a tendency to compared each schemes in terms of error protection coverage, performance impact, and energy consumption increase, that measure calculable with the Bench benchmark suit exploitation single-level cache. The results show that our theme has high error protection coverage of ninety seven while not virtual performance hit and additional energy consumption of DRAM, whereas ICR degrades overall system performance by around ten and will increase DRAM energy consumption by 2 hundredth, on average. Since our theme relies on spacial vicinity of programs, the error protection coverage are going to be completely different betting on cache sizes, cache levels, program behaviors, and also the range of cores. Thus, we have a tendency to experiment our theme with varied cache configurations. First, we have a tendency to use a two-level cache hierarchy in a very single-core processor, by dynamical cache associative and cache sizes, exploitation the SPEC2000 benchmark suit. The error protection coverage is slightly completely different betting on cache levels and sizes however show over ninety fifth and ninety six for the L1 and L2 caches, severally. Second, the projected theme is additionally simulated in a very multicore surroundings with SPEC2006 benchmarks. For the shared L2 cache in dual- and quad-core processors, our theme protects over ninety fifth and ninety six of tag bits of dirty cache lines, severally.

This paper is organized as follows. Section II shortly provides survey on standard error protection mechanisms and additionally discusses previous schemes projected to cut back vulnerability of cache recollections against transient errors. Section III details tag bits errors classified into four subgroups and standard ways elaborated downside Definition Section IV describes our projected cache design exploiting same tag bits. Section V presents our experimental

results for our theme and ICR. Finally, Section VI concludes this paper.

2. Issues Related to Conventional Methods

2.1 Reducing Cache Miss Penalty

Cache miss penalty is reduced by following techniques [1]:

a. construction caches

The performance gap between processors and memory leads the creator to the current question: ought to I create the cache quicker to stay pace with the speed of CPUs, or create the cache larger to beat the widening gap between the central processing unit and main memory though the idea of adding another level in the hierarchy is easy, it complicates performance analysis. Definitions for a second level of cache area unit not perpetually easy. The initial call is the size of a second-level cache. Since everything in the first-level cache is possible to be in the second-level cache, the second-level cache ought to be a lot of larger than the primary. If second-level caches area unit simply a touch larger, the native miss rate are high [2]. This observation conjures up style of immense second level caches—the size of main memory in older computers! One question is whether or not set associativity makes additional sense for second-level caches.

b. vital words first and early restart

Multilevel caches need further hardware to cut back miss penalty, however not this second technique. It's supported the observation that the central processing unit usually desires only one word of the block at a time [2]. This strategy is impatience: Don't stay up for the total block to be loaded before causation the requested word and restarting the central processing unit. Here area unit has two specific strategies: vital word first-Request the incomprehensible word first from memory and send it to the central processing unit as shortly because it arrives; let the central processing unit continue execution whereas filling the rest of the words in the block. Critical-word-first fetch is conjointly known as wrapped fetch and requested word 1st. Early restart-Fetch the words in traditional order, however as shortly because the requested word of the block arrives, send it to the central processing unit and let the central processing unit continue execution.

c. Giving priority to browse misses over writes

The simplest reply of this perplexity is for the browse miss to attend till the write buffer is empty. the choice

is to check the contents of the write buffer on a browse miss, and if there are no conflicts and the memory system is accessible, let the browse miss continue. Just about all desktop and server processors use the latter approach, giving reads priority over writes. The price of writes by the processor in a write-back cache will conjointly be reduced [3]. Suppose a browse miss can replace a unclean memory block. Rather than writing the dirty block to memory, and so reading memory, we tend to might copy the dirty block to a buffer, then browse memory, and then write memory. This manner the central processing unit browse, for that the processor is maybe waiting, can end sooner. Similar to the scenario on top of, if a browse miss happens, the processor will either stall till the buffer is empty or check the addresses of the words within the buffer for conflicts.

d. Merging write buffer

If the buffer is full and there's no address match, the cache (and CPU) should wait till the buffer has an empty entry. This optimization uses the memory additional with efficiency since multiword writes are unit sometimes quicker than writes performed one word at a time. The optimization conjointly reduces stalls due to the write buffer being full. Assume we tend to have four entries within the write buffer, and every entry might hold four 64-bit words [3]. While not this optimization, four stores to successive addresses would fill the buffer at one word per entry, even supposing these four words once unified precisely match within one entry of the write buffer.

e. Victim Caches

One approach to lower miss penalty is to keep in mind what was discarded in case it is required once more. Since the discarded knowledge has already been fetched, it will be used once more at tiny price. Such "recycling" needs a little, totally associative cache between a cache and its refill path.

2.2 Reducing Miss Rate

Miss rate is reduced by following technique:

a. mistreatment larger block size

The simplest manner to cut back miss rate is to increase the block size. Larger block sizes can cut back required misses. This reduction happens as a result of the principle of neighborhood has 2 components: temporal neighborhood and spacial neighborhood. Larger blocks make the most of spacial neighborhood.

b. mistreatment larger caches

The obvious downside is longer hit time and better price. This method has been particularly widespread in off-chip caches: the dimensions of second or third level caches in 2001 equals the dimensions of main memory in desktop computers from the primary edition of this book, solely a decade before.

c. Higher associativity

The first is that eight-way set associative is for sensible functions as effective in reducing misses for these sized caches as totally associative [4]. You will be able to see the distinction by examination the 8-way entries to the capability miss column, since capability misses are unit calculated mistreatment totally associative cache. The second observation, known as the 2:1 cache rule of thumb and located on the front within a direct-mapped cache of size N has regarding constant miss rate as a 2-way set associative cache of size $N/2$. This control for cache sizes lower than 128 computer memory unit.

d. manner prediction and pseudo associative caches

Another approach reduces conflict misses and nevertheless maintains the hit speed of direct mapped cache. In way-prediction, further bits are unit unbroken in the cache to predict the set of the next cache access [4]. This prediction means that the multiplexor is about early to pick the required set, and solely one tag comparison is performed that clock cycle. A miss ends up in checking the opposite sets for matches in resulting clock cycles.

e. Compiler optimizations

Thus far our techniques to cut back misses have needed changes to or additions to the hardware: larger blocks, larger caches, higher associativity, or pseudo-associativity. This final technique reduces miss rates with none hardware changes. This magic reduction comes from optimized software—the hardware designer's favorite solution! The increasing performance gap between processors and main memory has impressed compiler writers to scrutinize the memory hierarchy to ascertain if compile time optimizations will improve performance. All over again analysis is split between enhancements in instruction misses and enhancements in knowledge misses.

2.3 Reducing Hit Time

Hit time is reduced mistreatment following technique:

a. Tiny and easy cache

A long portion of a cache hit is mistreatment the index portion of the address to browse the tag memory and so

compare it to the address. It is conjointly vital to keep the cache tiny enough to match on the same chip as the processor to avoid the time penalty of going off-chip. The second suggestion is to stay the cache straightforward; resembling mistreatment directs mapping [5]. A main advantage of direct-mapped caches is that the designer will overlap the tag check with the transmission of the info. This effectively reduces hit time.

b. Avoiding address translation throughout categorization of the cache

The guideline of creating the common case quick suggests that we tend to use virtual addresses for the cache, since hits area unit a lot of additional common than misses. Such caches area unit termed virtual caches, with physical cache used to determine the normal cache that uses physical addresses. Page level protection is checked as a part of the virtual to physical address translation, and it should be implemented regardless of what. one more reason is that each time a method is switched, the virtual addresses refer to completely different physical addresses, requiring the cache to be flushed [6]. A third reason why virtual caches area unit not additional widespread is that in operation systems and user programs could use 2 completely different virtual addresses for the same physical address.

c. Pipelined cache access

The final technique is just to pipeline cache access so the effective latency of a primary level cache hit will takes four clocks. This split will increase the quantity of pipeline stages, resulting in larger penalty on mispredicted branches and additional clock cycles between the problem of the load and also the use of the info.

d. Trace cache

A challenge in the effort to realize instruction level correspondence on the far side four directions per cycle is to offer enough directions each cycle while not dependencies. Trace caches have a lot of additional difficult address mapping mechanisms, because the addresses are not any longer aligned to power of two multiple of the word size [6]. They need alternative edges for utilization of the knowledge portion of the instruction cache. Trace caches store directions solely from the branch entry purpose to the exit of the trace.

3. Conventional Caches

3.1. Web Cache

Web pages and pictures from previous responses of internet servers are hold on in internet caches been utilized by internet proxy servers and internet browsers. The quantity of info needed to transmit across the network is reduced by internet caches as info hold on in the cache antecedently will usually be reused. This method ends up in the reduction of process necessities and information measure of the internet server that jointly aids in the improvement of the responsiveness for internet users [7]. The technology of internet caching has been wide used to decrease the perceived network latencies of users and to improve the internet infrastructure performance. The major technique used in internet caching that endeavors (from a network of proxies) to serve user internet requests between the top users and the internet server (that hosts the requested object's original copies) is thought as proxy caching [8,9].

3.2. Hybrid Cache

Performance of the chip in terms of process speed and consumption of power will be improved victimisation the design of hybrid cache. The structure of every memory has least satisfies the Average operation Time (AMAT) underneath a given area condition. The design of the hybrid cache that compose of the static random access memory (SRAM) and magnetic random access memory (MRAM) shows that sixteen.9% reduction in AMAT and fifteen.2% of reduction in power compared with the cache that is composed of the solid SRAM. The SRAM and the dynamic random access memory (DRAM) structure represents thirty three.0% reduction in the consumption of power whereas that of SRAM and part modification random access memory (PRAM) shows a potential in the reduction of space and consumption of power as a results of the RAM's (SRAM and PRAM) high density [10,11].

3.3. Non-Uniform Cache Design

The non-uniform cache design (NUCA) is a brand new approach that is being mentioned for the future of many-core processors. These NUCA recollections are an different to increase the performance of processors that have a terribly great amount of process cores and need a heap of information measure. In the case of shared memory, one will purpose the access latency as the main drawback for the today uniform cache architectures. The NUCA recollections will cut back the latency of knowledge access, increasing measurability, thus, these recollections have a nice potential for increasing performance on multi-core and many-core processors. Nowadays, multiple levels of cache

recollections are organized in a few distinct levels. Typically, every lower level includes and replicates the content of on top of levels, thereby reducing access time to levels nearer to the main memory. For future technologies, giant memory cache among the chip, with a single information access time isn't acceptable, since issues connected to the wire-delay within the chip are increasing. This is the main argument for creation of new non-uniform cache design. Besides the wire-delay drawback, we will cite the necessary behavior of potential parts failures on time, that next generation processors shall have [11]. Thus, some isolation or redundancy on the memory modules and interconnection will guarantee that if any failure happens the opposite parts can keep working. The main conception involving these new non-uniform cache architectures is that some information can reside nearer to the processor, thus, these information can have some quick access times. This way, close to information are accessed quicker than information that is distant from the processor. Moreover, issues related to the quantity of ports and hardware failure will be part resolved by a non-centralized design.

3.4. Associative Mapping

In this form of mapping the associative memory is employed to store content and addresses each of the memory word. This allows the position of the any word at anyplace within the cache memory. It's thought-about to be the quickest and also the most versatile mapping type.

3.5. Direct Mapping

In direct mapping the RAM is formed use of to store information and a few is hold on within the cache. AN address house is split into 2 components index field and tag field. The cache is employed to store the tag field whereas the remainder is hold on within the main memory. Direct mapping's performance is directly proportional to the Hit magnitude relation. After the firm analysis we conclude direct mapping terms to be the most efficient method in terms of area, power and delay which has simulated in the below specified architecture.

Target Device: xc3s250e
Target Package: tq144
Target Speed : -4
Mapper Version: Spartan3E

Table.1. Performance analysis for the existing and proposed architecture

	Base paper Existing	Base paper proposed
LUTs	425 out of 63,400	71 out of 4896
BELS	437	90
Delay		
Minimum input arrival time before clock	5.714ns	6.773ns
Maximum output required time after clock	1.102ns	4.283ns
Power	0.165 Watts	0.080 Watts
Memory	506756 Kilobytes	262724 Kilobytes
Flip Flops/Latches	11	43
	Base paper Existing	Base paper proposed
Total REAL time to PAR completion	23 secs	8 secs
Total CPU time to PAR completion	22 secs	7 secs

4. Performance Analysis

Completely different strategies and techniques for cache optimization that have been enforced in the recent past have been analyzed. we tend to compare most of those strategies and techniques for completely different parameters and gift them in Table one. These strategies are compared with every different on bases of the listed parameters. Miss rates (MR), Miss penalty (MP), Hit time (HT), Hit Rates (HR), Power Consumption (PC), interval (AT), Cost, Complexity, Cycle Time (CT). All of these strategies have some blessings and disadvantage. Table one summarizes of these blessings and downsides. Larger block size of cache is that the easiest method to cut back the miss rate but they can manufacture longer hit times alongside higher price. Larger block size cache conjointly increase the conflict misses. In distinction smaller block size cache reduces cache hit times. Multi-level caches are quite economical in up the overall system performance but hierarchy style of multi-level caches is extremely complicated.

5. Conclusion

This paper describes the varieties of cache and however they will improve the system performance

and speed in terms of information accesses. Elaborated steps concerned in information transfer between the cache and memory were conjointly mentioned. Cache entries are created once cache lines are derived from the most memory to the cache that includes the derived information along with the requested memory location. The hit rate will be alive of the cache's effectiveness for a given formula or program.

When an information transfer is required from the main memory abundant additional slowly than the cache itself, the scan misses delay execution whereas the write would possibly take place while not such penalty, since execution may be continuing by the processor as information – in the background – is derived to the main memory. The foremost concern of the cache style is that the consumption of energy. The applying victimization the cache determines the performance of the cache design. Internet pages and pictures from previous responses of internet servers are held on in internet caches been utilized by internet proxy servers and internet browsers. The quantity of info needed to transmit across the network is reduced by internet caches as info held on in the cache antecedently will usually be reused. This method results in the reduction of process necessities and information measure of the internet server that jointly

References

- [1] Jeongkyu Hong, Jesung Kim, and Soontae Kim, Member, IEEE, "Exploiting Same Tag Bits to Improve the Reliability of the Cache Memories", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 23, NO. 2, FEBRUARY 2015
- [2] "An Overview of Cache Optimization Techniques and Cache", Markus Kowarschik and Christian Wei; Fakultät für Informatik Technische Universität München, Germany.
- [3] "A Trace Cache Microarchitecture and Evaluation", Eric Rotenberg, Steve Bennett; IEEE TRANSACTIONS ON COMPUTERS, VOL. 48, NO. 2, FEBRUARY 1999.
- [4] "Investigation of Impact of Victim Cache and Victim Tracer on a Fully Associative Disk Cache", R. Pendse, N. Kushanagar, U. Walterscheidt; Department of Electrical Engineering, The Wichita State University, 1845 North Fairmount.
- [5] "David A. Patterson, John L. Hennessy. "Computer organization and design: the hardware / software interface". 2009. ISBN 0-12-374493-8, ISBN 978-0-12-374493-7 "Chapter 5: Large and Fast: Exploiting the Memory Hierarchy". p. 484.
- [6] "Zeng, D., F.-Y. Wang, and M. Liu, Efficient web content delivery using proxy caching techniques. Systems, Man, and Cybernetics, Part C: applications and Reviews, IEEE Transactions on, 2004. 34(3): p. pp. 270-280 (3).
- [7] Rabinovich, M. and O. Spatschek, Web caching and replication. 2002: Addison-Wesley Longman Publishing Co., Inc. 361.
- [8] Takase, T., et al., A web services cache architecture based on xml canonicalization, in In Proceedings 11th Int. World Wide Web Conf. (Poster Paper) 2002, HI: Honolulu.
- [9] Brooks, C., et al., Application-specific proxy servers as http stream transducers, in In Proceedings 4th World Wide Conference 1995. p. pp. 539–548.
- [10] Lee, S., J. Jung, and C.-M. Kyung. Hybrid cache architecture replacing SRAM cache with future memory technology. in Circuits and Systems (ISCAS), 2012 IEEE International Symposium on. 2012. IEEE.
- [11] Aggarwal, N., et al., Isolation in Commodity Multicore Processors. Computer, 2007. 40 (6)(6): p. pp. 49-59.

