VOL. 11, NO. 5, MARCH 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A LITERATURE SURVEY ON CPU CACHE RECONFIGURATION

S. Subha
SITE, Vellore Institute of Technology, Vellore, India
E-Mail: ssubha@rocketmail.com

ABSTRACT

CPU caches are designed with fixed number of sets, associativity and line size. Some methods are proposed in the literature for certain combinations of variable cache sets, variable cache ways, variable cache lines. This paper gives a survey of significant results in this regard.

Keywords: cache reconfiguration, variable cache lines, variable cache sets, variable cache ways.

1. INTRODUCTION

A CPU cache is denoted by the tuple (C, k, L) where C is the capacity, k the associativity and L the line size. Caches are of three kinds viz. direct mapped, set associative and fully associative. A line is placed in fixed position in direct mapped cache. A line can occupy any of w ways in w-way set associative cache. A line can occupy any of n blocks in fully associative cache of n blocks. [3, 8]. The number of cache sets, line size and associativity is fixed for cache. A direct mapped cache can be thought of as one-way set associative cache while fully associative cache of n blocks can be thought of as cache with one set and n blocks. The three cache parameters can be visualized as the three dimensions of cache configuration. This is shown in Figure-1. The dimensions can be changed based on the cache view. For given cache of line size L bytes with associativity k, and S sets, the total capacity C=SkL bytes. The three cache parameters are usually powers of two

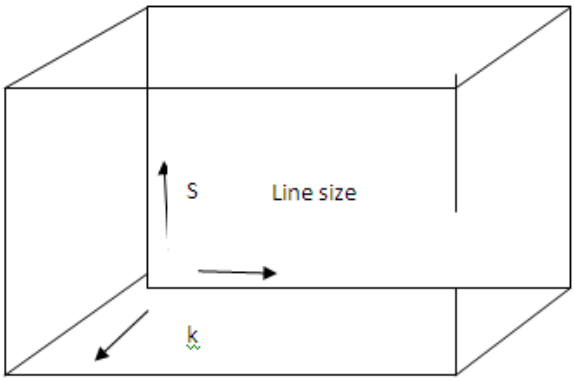

Figure-1. The three dimensions of cache. Here S is the number of cache sets k set associativity.

Each main memory address is mapped to cache line during program execution. A memory address a has the tag, index and block offset as shown in Figure-2. An address a is mapped to set a mod S with tag value of a div S in cache of S sets.

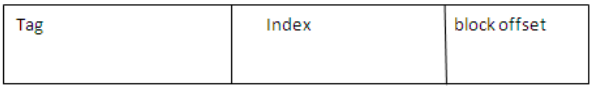

Figure-2. Address structure for cache mapping.

If a line is found in cache, it is called cache hit else it is called cache miss. Cache misses are of three kinds-cold, capacity and conflict. A miss for first occurrence of an address is cold miss. The capacity misses are the difference between the misses in given cache and misses in fully associative cache of same size. The miss caused due to the cache set being full in direct mapped and set associative caches is called conflict miss. The conflict miss for fully associative cache is zero or undefined. The least recently used algorithm is used to replace cache lines on conflict in set associative and fully associative caches. [3, 8].

Usually a system has several cache levels. The cache closest to the CPU is level one followed by level two, level three etc. cache levels. The cache size grows with the level number. The cache access time increases with the cache level. Caches are of two kinds-inclusive and exclusive. If a cache line is present in all cache levels, it is inclusive cache. If cache line is present in only one cache level, it is called exclusive cache. A two type data cache model was proposed for two level cache in [20] in which the cache is inclusive or exclusive based on hit patterns.

The cache performance is measured in terms of access to cache lines. The ratio of total number of hits to total number of cache accesses is called hit ratio. The ratio of total number of misses to total cache accesses is called miss ratio. For given memory hierarchy, the average memory access time denoted by AMAT is used as cache performance metric. For given trace of R references for two level inclusive cache hierarchy if the hits in level one and level two are h_1, h_2 respectively, t_1, t_2, t_{12} are the level one access time, level two access time, transfer time between level one and level two, m the miss penalty, the AMAT is given below.

$$AMAT = \frac{1}{R}\left(h_1 t_1 + h_2\left(t_1 + t_2 + t_{12}\right) + \left(R - h_1 - h_2\right)m\right) \quad (1)$$

The first term in (1) is the access time for level one hits. The second term is the access time for level two hits. This involves accessing level one cache, level two cache and transfer the line from level two to level one cache. The third term is the access time during cache miss. The AMAT is affected by cache configuration. This paper gives survey of the certain algorithms and models proposed to reconfigure the cache by adjusting cache parameters.

The paper is organized as follows. Section 2 gives survey on reconfiguring cache sets, section 3 gives survey on reconfiguring cache lines and section 4 gives survey on reconfiguring cache sets Section 5 gives the conclusion followed by references.

2. RECONFIGURE CPU CACHE SETS

The number of cache sets is fixed at cache design. The set size is fixed at design phase. The cache set term is applicable to direct mapped and set associative caches. The set number for any address a is obtained from the function a mod S for cache of S sets. If line is present in cache set, it is accessed. Else replacement algorithm decides the location of the line. Usually the least recently used (LRU) line is replaced in set associative cache [3, 8]. Usually all the cache sets are in high power mode during cache operation. Various replacement algorithms are proposed in literature. The number of cache sets cannot be altered physically during cache operation. However various algorithm/models are proposed in literature to have variable cache set architecture at logical level.

The author in [1] proposed column associative cache. This cache extends the mapped set to an alternate set in direct mapped cache. A line is placed in mapped set if possible. Else, the most significant bit of mapped set is negated to obtain alternate set. The line is placed in alternate set if possible. A hash bit is maintained per set to indicate if the line is mapped or alternate line. The authors reported improvement in miss ratio. The entire cache is in operational mode in this model.

The author in [16] proposed cache reconfiguration using gated vdd. In [16] the number of index bits and their positions decide the active sets. A block's placement changes in [16] based on the selected sets. The cache size need not be power of two in the proposed model. The dynamic resizing in [16] depends on the miss ratio.

The author in [9] proposed model in which sets are associated with conflicting set. In [9], the associated cache sets for conflicting set is defined to reduce conflicts to a set. These sets are probed during address mapping. An average reduction of 13% in miss rate was observed for simulations on SPEC2006 benchmarks in this method.

The author in [12] proposed cache architecture in which certain fixed number of sets is present for the cache which can be extended by suitably selecting from the tag bits. The set and way management cache architecture for run time management [SMART cache] has variable sets and ways architecture. A decision tree based macine learning model adapts the cache set size. In [12] the sets are turned off during non-use. However the dirty lines need to be updated in cache system. These lines are remapped in future access. A performance degradation of 2% was observed with improvement in energy-delay product.

The author in [24] proposed cache set reconfiguration. The proposed model starts with one set and doubles the number of cache sets on conflicts. The tag field size varies among the sets. There is no remapping of cache lines. However the number of cache sets is power of two. In the proposed model, the index bits are not masked, a block's placement is fixed, the number of active sets varies but the total number of sets is a power of two, a new set is included in the active sets based on the conflict miss in the active sets. An improvement in AMAT was observed.

The author in [27] proposed cache set reconfiguration. The cache sets have register. Each register has number of bits equal to number of cache sets. On occupancy, the set number is set to one in the cache set. A logical AND with set number and logic one enables cache ways The number of cache sets is increased by one on all sets full condition. The number of cache ways per set is unaltered in this model. On access all the ways containing the lines in mapped set are searched. The replacement policy is to replace the LRU of occupied ways of same set on conflict. If a new set is accessed on cache conflict, the last set is used to place the line using LRU policy. A power saving of 12% with AMAT degradation of 16% was observed.

The cache set reconfiguration is interesting topic. A model configuring the sets at physical level with minimal power consumption and improvement in AMAT is topic of future research.

3. RECONFIGURE CPU CACHE LINES

The cache line is fixed size during cache design. Variable cache line size is interesting research topic. Once the line size is fixed, the associativity for given number of cache sets can be designed for fixed number of cache sets. The author in [3] has used profiling to decide on optimal cache line size. The cache line size is power of two in this proposed model. The miss ratio was chosen as performance measure in this study.

The authors in [4] propose cache line adjustment based on observed application access to the line in hardware. However the cache line size in this method is always power of two. The proposed hardware algorithm keeps track of line access and adjacent line access. Based on this activity the line size is incremented or decremented. The proposed method was simulated on chosen SPEC92 and SPEC95 benchmarks. The proposed method resulted in performance improvement in terms of

miss ratio for certain benchmarks. The amount of memory traffic significantly decreased in all benchmarks compared to fixed line size case. It is recommended that the best strategy for applying adaptivity is to use large initial line size and have the adaptivity decrease as it is needed.

The author in [17] proposes method to decide on the optimal line size for various variables for matrix multiplication algorithm using optimization techniques. The concept is extended to loops and programs [18, 19] with certain variable restrictions. The line size in these cases need not be power of two. A hardware lookup table keeps track of line size of various variables in this design. This design can be extended to accommodate global and pointer variables in programs as future work.

The design goal to have variable cache line size to improve cache performance with minimal hardware support is topic of future research.

4. RECONFIGURE CPU CACHE WAYS

The number of cache ways in set associative cache is fixed at design time. Not all cache ways are occupied during program execution.

The author in [7] proposed cache design to disable a subset of cache ways in set associative cache during periods of modest cache activity while the full cache may remain operational for more cache-intensive periods. A small performance degradation with energy savings was observed in this model.

The authors in [10] provide model to adjust n-way set associative cache to be one way. A performance degradation with energy saving was observed in this model.

The cache model proposed in [16] provides for variable cache ways using gated vdd. The number of cache ways per set is fixed in this proposed model during reconfiguration.

The author in [5] proposes method to utilize the cache ways among the sets based on static profiling.Each set in the proposed design basically has different associativity to maximize the total performance for the cache size or reduce the power consumption. The proposed architecture is simulated on SimpleScalar simulator and tested on several Spec2000 Benchmarks. The results show on average 2% reduction in the miss rate at the high-performance mode and up to 43% reduction of the power consumption at low-power mode. The design is given below.

a) The cache sets are divided into two parts - fixed and variable. The fixed size is equally distributed among the sets. The variable part is shared among sets with more associativity requirement.
b) The program is runwith fixed cache sets and whole cache. The additional ways required for each set is determined. This is used for cache design.

The V-way cache proposed in [15] proposes cache model with variable cache ways. The number of tag store entries is increased relative to number of data lines making global replacement feasible maintaining constant hit time latency of set associative cache. A performance improvement of reduction of 13% on miss rate on sixteen SPEC2006 benchmarks was observed.

The author in [23] uses the concept of registers as cache ways. The cache is considered as set of registers in this design. The number of cache ways per set is variable in this design. The proposed model was simulated with SPEC2K benchmarks. A performance improvement of 3% in AMAT was observed.

The authors in [11] propose a cache model allowing sharing of cache ways among a pair of cache sets resulting in energy saving and negligible performance penalty.

The author in [21] proposed extending cache ways to adjacent sets in circularly arranged cache sets. The cache ways are extended half way to sets on either side making 2w-way set associative cache in w-way set associative cache. An owner bit per cache way indicates the line's ownership to mapped set or adjacent set. Theproposed model was simulated with SPEC2K benchmarks. A performance improvement of 5% over traditional cache and 3% over 2w-way set associative cache of same capacity for chosen cache parameters was observed.

The author in [22] proposed model to extend the cache ways to any other set instead of adjacent sets as in [21]. An other_set entry per cache set indicates the alternate set. An owner bit per cache way indicates the line ownership. Each set can be associated only with one more set in this model. The proposed model was simulated with SPEC2K benchmark with AMAT improvement of 2.5%. over traditional set associative cache.

The author in [27] proposed model to extend the concept of way sharing among sets. Any way can be occupied for any set in this architecture. A register per cache set is maintained. The number of bits in register is equal to the number of cache sets. A logic AND with logic one and set bit is used to enable the cache way. The proposed model is simulated with SPEC2K benchmarks with performance degradation of 16% with power saving of 12% were observed in this design.

5. CONCLUSIONS

CPU cache reconfiguration has been done per cache parameter widely. Cache reconfiguration affects the power consumed in the cache operation. The goal is to minimize the power consumption with performance improvement. However both the goals are not usually achievable. Hence one has to compromise on performance for power savings. The design goal can be stated thus.

minamat by cache reconfiguration
subject to

min power consumption

The power and performance improvement have been reported in literature for set associative caches [25]. A cache model for fully associative cache with power saving and no change in performance is reported in [26]. Many models have been proposed for power/performance improvement. The authors in [6] examine performance and power trade-offs in cache designs and the effectiveness of energy reduction for several novel cache design techniques targeted for low power. The detailed mathematical model for power dissipation is discussed in [13]. The authors in [14] propose a method to access one cache way for energy saving in the mapped set in set associative cache. The way is predicted using MRU policy. An improvement in energy consumption is reported in this model.

A few research designs involving combination of cache parameters has been suggested in literature. The gated Vdd involves reconfiguring the cache ways and cache sets [16].

To conclude an ideal cache configuration would be to suggest cache line size for given cache size in sets determining the associativity that minimizes the AMAT. The reconfiguration proposed so far involves mostly addition of active hardware components during cache operation. This increases the power consumption of the cache usually. Models that do not increase the power consumption of cache reconfiguration are topic of future work.

REFERENCES

[1] Agarwal A and Steven D. Pudar. 1993. Column Associative Caches A Technique for Reducing the Miss Rate of Direct Mapped Caches.Proceedings of the 20th annual international symposium on Computer architecture. pp. 179-190.

[2] Alan Jay Smith, Cache Memories, Computing Surveys. 14(3), September1982,

[3] Alan Jay Smith, Line (Block) Size Selection in CPU Cache Memories, IEEE Transactions on Computers, Vol. C-36, No.9, September. 1987, pp. 1063 -1075.

[4] Alexander V. Veidenbaum, Weiyu Tang and Rajesh Gupta. 1999. Adapting Cache Line Size to Application Behaviour. Proceedings of the 13th International conference on Supercomputing. pp. 145-154.

[5] AlyR.E., NallamilliB.R., Bayoumi M.A. 2003. Variable-way Set Associative Cache Design for Embedded Systems Applications. Proceedings of IEEE Symposium on Circuits and Systems.

[6] C.Su, A.Despain. 1995. Cache Design tradeoffs for power and performance optimization: a case study. Proceedings of international symposium on Low power electronics and design. pp. 63-68.

[7] D. Albonesi. 1999. Selective Cache Ways: On Demand Cache Resource Allocations, Proceedings of ISCA. pp. 248-259:

[8] David.A. Patterson, John L Hennessey. 2003. Computer System Architecture,A Quantitative Approach, 3rd Edition, Morgan Kaufmann Publishers Inc.

[9] Dyer .Rolan, B.B. Fraguela, R.Doallo. 2009. Adaptive Line placement with Set balancing Cache. Proceedings of MICRO, December.

[10] Hsin-Chum Chen and Jen-ShiunChimg. 2001. Designing of An Adjustable way set associative cache, Proceedings of PACRIM.

[11] JanrajC. J., KalyanT.V., Warrier.T, Mutyam M. 2012. Way Sharing Set Associative Cache Architecture. Proceedings of VLSID. pp. 251-256.

[12] K.T.Sundararajan, Timothy M.Jones, Nigel Topham. 2011. Smart Cache: A Self Adaptive Cache Architecture for Energy Efficiency. Proceedings of International Conference on Embedded Computer Systems. pp. 41-50.

[13] Ko U, Balsara P.T. 1995. Characterization and Design of A low power High performance Cache Architecture. Proceedings of International symposium on VLSI technology and Systems and applications. pp. 235-238

[14] Koji. Inoue, T.Ishihara and K.Murakami. 2000. A High-Performance and Low-Power Cache Architecture with Speculative Way-Selection.IEICE Transactions on Electron. E83-C(2): 186-193.

[15] Moinuddin K. Querishi, David Thompson, Yale N Patt. 2005. The V-Way Cache: demand based associativity via global replacement.Proceedings of ISCA. 33(2): 544-555.

[16] Se-Hyun Yang, Michael D.Powell, BabakFalsafi, T.N.Vijaykumar. 2002. Exploiting Choice in Resizable Cache Design to Optimize Deep-Submicron Processor Energy-Delay. Proceedings of HPCA. pp. 151-161.

[17] S. Subha and Weijia Shang. 2006. Variable Block Size Architecture for Matrix Multiplication, Proceedings of Obcom. pp. 187-191.

[18] S. Subha. 2008. Variable Block Size Architecture for Loops, Proceedings of the Fifth International Conference on Information Technology: New Generations. pp. 1144-1145.

[19] S. Subha. 2009. Variable Block Size Architecture for Programs. ITNG 2009, pp. 1640-1641.

[20] S. Subha. 2009. A Two-Type Data Cache Model. Proceedings of 2009 IEEE International Conference on Electro/Information Technology. pp. 476-481.

[21] S.Subha. 2013. An Algorithm for Variable Cache Ways, Proceedings of ICATE 2013.

[22] S.Subha. 2013. A Cache Architecture, Proceedings of ICACT.

[23] S. Subha. 2013. An Algorithm for Variable Cache Ways. International Journal of Computer and Communication Engineering. 2(5): 557-559.

[24] S. Subha. 2014. An Energy Saving Cache Algorithm. 2014 International Conference on Computational Science and Technology (ICCST).

[25] S.Subha. 2014.An Energy Saving Set Associative Cache Algorithm with Improved Performance. JATIT. 62(3).

[26] S.Subha. 2015. A Fully Associative Cache Architecture, JATIT. 73(3).

[27] S.Subha. 2015. A Variable Cache Set Architecture, IJAER. 10(20): 41350-41352.