Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, Volume 23, No.1, ISSN 2304-1382
ISBN 978-3-901509-91-9, CDROM version, Ed. B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2012
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2012

PERFORMANCE-ORIENTED DESIGN OF FAULT TOLERANT CACHE ARCHITECTURES

NOVAC, O[vidiu] C[onstantin]; VARI - KAKAS, S[tefan];
NOVAC, C[ornelia] M[ihaela] & GORDAN, C[ornelia] E[milia]

Abstract: *In this paper, we present aspects regarding the possibility of maximizing the performance/cost ratio in the cache memory design. A cache is a component that improves performance by transparently storing data so that future requests for that data can be served faster. Caches provide for efficient read/write access to memory, and their reliability is essential to assure dependable computing. Usually the cache design goal is to find such an architecture that maximizes the hit ratio. This paper offers the designer a computer simulation based method, taking into account classical performance metrics and reliability aspects.*
Keywords: *cache, hit ratio, block size, overhead, fault tolerance, simulation*

1. INTRODUCTION

In every technical design a crucial problem is the maximization of the performance/cost ratio. This is also the case in computer system design, particularly in that of a cache memory, part of a memory hierarchy [1], [2], [3], [4]. A difficult problem is to find the best architecture since it depends not only on the hardware organization, but also on the memory reference traces of the executed program, which cannot be predicted. Usually the design is supported by computer simulation, which permits the run of different traces, with variable parameters. In order to compare different architectural solutions certain parameters, like cost, must be assumed to be constant. We will concentrate on a fault tolerant cache architecture, where the overhead due to redundant elements has to be considered, too [5], [6].

2. FAULT TOLERANT CACHE AND PERFORMANCE METRICS

Fig. 1 shows a cache architecture with its access logic. The connections of the cache to the CPU and the main memory are demonstrated. The CPU provides the write and read signals and the physical address of the accessed data. The cache presents an associative organization in rows, each row containing a block of data. A cache hit is produced if there is any address in the tag part, which is identical with the block address field of the issued physical address. The type of associativity (fully associative, n-way set associative, direct mapping), the number of rows and the block size have direct influence on the hit ratio, i. e. the percent of cache hits from the total number of memory accesses in a given period:

$$\text{Hit ratio} = \frac{\text{Nr. hit}}{\text{Nr. hit} + \text{Nr. miss}} \qquad (1)$$

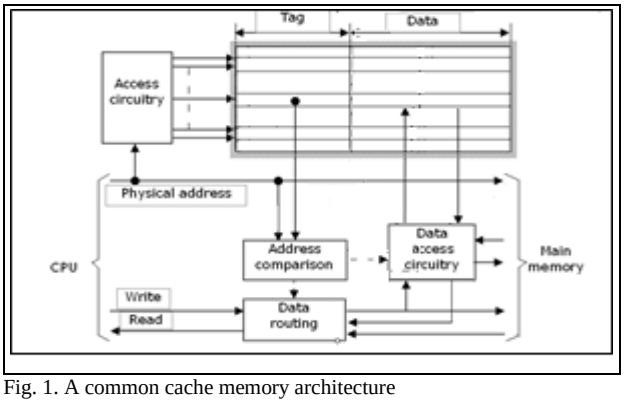

Fig. 1. A common cache memory architecture

We present a cache structure with access circuitry in order to observe the places where the redundant elements have to be introduced. Fig. 2 demonstrates the general organization of a fault tolerant cache memory [7], [8]. The fault tolerance is assumed to be implemented in the data area, based on information redundancy with the use of an error correcting code. This creates an additional block overhead in the data area. (We assume that the overhead generated by the check bits control circuitry is negligible.)

The goal by fault tolerance is to enhance the reliability of the memory, i. e. to increase the number of tolerated faults per block [9]. This however implies a great number of control (check) bits, leading to a high block overhead [10]. The block overhead can be calculated on an equivalent cell area as the sum of the area of functionally different bits:

$$\text{Block overhead} = \text{tag} + \text{status} + \text{replacement} + \text{control} \qquad (2)$$

In this equation the overhead due to the tag bits is also considered, being that this kind of data is not included in the common definition of the cache memory capacity. The status and replacement fields belong to each row, expressing the validity of the block and the history of access (to be used in the replacement algorithm), respectively.

The problem is to find the best architecture that leads to a maximum of the main performance metrics: hit rate and utilization ratio. The utilization ratio can be defined as in [11]:

$$\text{Utilization ratio} = \frac{\text{Data area}}{\text{Data area} + \text{Block overhead}} \qquad (3)$$

Fig. 2. A cache memory architecture with fault tolerant capability in the data area

In order to compare different architectural solutions, it is necessary to assume some constant parameters:

- Address word length (32 bits)
- Memory bus width (64 bits)
- Block information bits (status + replacement)
- Replacement algorithm (LFU ï Least Frequently Used)
- Consistency algorithm (write through with write allocate)

The variable parameters in the simulation are:

- Mapping (fully associative, n-way set associative, direct mapping)
- Block size
- Block control bits

The size of a block has direct influence on the traffic ratio, which however should be minimized:

$$\text{Traffic ratio} = (1 - \text{Hit ratio}) \times \frac{\text{Block size}}{\text{Bus size}} \qquad (4)$$

It is quite difficult to find the optimal fault tolerant cache architecture via analytical calculus, but it is possible to tend to it by means of computer simulation (for example in [12], [13], [14]).

For cache simulation we used ourown program, developed in Microsoft Visual C++ [15]. The main user interface is shown in Fig. 3.

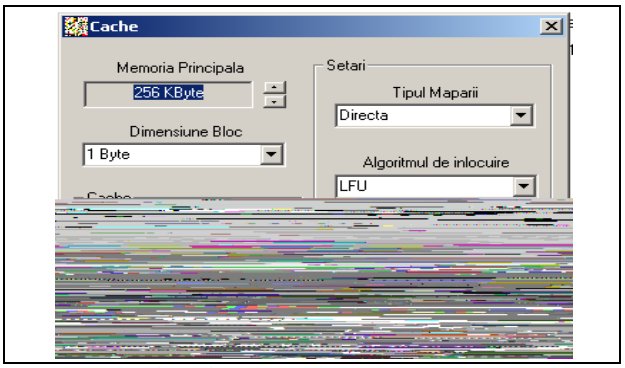

Fig. 3. Main interface of the cache simulator program

Our simulator program can be used for multiple settings of the cache memory. The program allows the configuration of multiple cache levels in a memory hierarchy [16].

The first setting is to choose the capacity of the main memory, which can range from 256 KB to 1 GB.

Another parameter to be configured is the number of levels of the memory hierarchy. The hierarchy can have one, two, three or four cache levels.

The next parameter of the program to be configured for each level is the block size which can be: 1 byte, 2 bytes, 4 bytes or 8 bytes.

The mapping function is another parameter to be set and it can be: fully associative, set associative or direct mapping.

For the replacement algorithm we can choose between LRU (Least Recently Used), LFU (Least Frequently Used), FIFO (First In First Out), or random.

The writing policy is another parameter to be configured. We can choose between write through or write back.

In Fig. 4 the hit ratio and the miss ratio can be seen as the result of a simulation with the following parameters: main memory of 1 GB, one cache level, direct mapping, LFU replacement algorithm, write through policy, block size of 8 bytes. We used a CEXP memory trace exhibiting a strong random behavior.

Fig. 4. Results of the cache simulator program

With our simulator it was possible to find a desired cache organization related to performance metrics for different memory access traces, maintaining the mentioned constant parameters and varying the others.

3. RESULTS

We performed several experiments with our cache simulator software, with some of the results presented below. In all experiments the same number of control bits per block was assumed.

Experiment 1.
For this simulation we chose the following parameters: 1 MB main memory, one cache level, direct mapping, 8 bytes block size, LFU replacement algorithm, write through policy, Wave trace file (derived from electromagnetic equations solving program). The results for a cache capacity of 8 KB are presented in Fig. 5.

Cache size (KB)	Hit ratio (block size 1 byte)	Hit ratio (block size 2 bytes)	Hit ratio (block size 4 bytes)	Hit ratio (block size 8 bytes)
8	0.6076	0.6093	0.6102	0.7570
16	0.6079	0.6096	0.6108	0.7578
32	0.6163	0.6181	0.6193	0.7651
64	0.6166	0.6184	0.6196	0.7660
128	0.6166	0.6184	0.6196	0.7660

Tab. 4. Comparative results of simulation experiments

Tab. 4 summarizes the results of the simulations accomplished for different block sizes. It is interesting to note that there is a significant jump in the hit ratio at the block size of 8 bytes.

Fig. 8 shows the hit rate versus cache size for the simulations presented in Tab. 4. It can be observed that the hit rate increases with block size until a threshold value is reached. Starting from this point, the hit rate does not change even if the cache size is increased, it is therefore not necessary to invest in larger cache memory, but better performance will be obtained if we work with a larger block size cache.

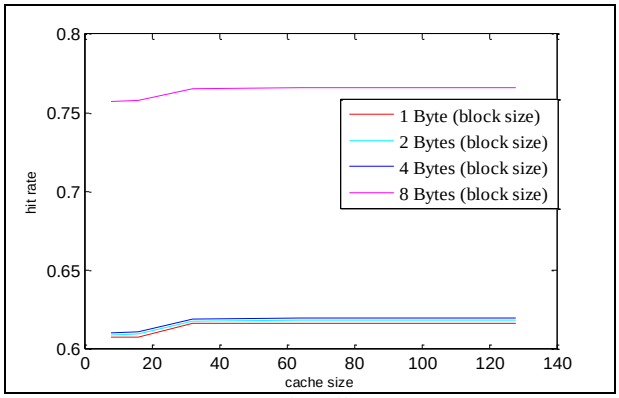

Fig. 8. Hit rate versus cache size for different block dimensions

4. CONCLUSIONS

Our work attempts to offer the designer of a cache memory architecture a computer simulation based method, taking into account classical performance metrics and reliability aspects.

The presented method can be useful in fault tolerant microprocessor system design. The simulation was performed using our own program, and in our future research activity we intend to develop the method as a software tool for complex memory system design.Theoretical studies regarding the development of best effort architectures follow two performance indicators. In this paper we present a software product currently pursuing one of these indicators. In the next phase of our research we will report the implementation of both performance indicators via a simulator.

5. ACKNOWLEDGEMENTS

This This work was co-financed from the European Social Fund through Sectorial Operational Programme Human Resources Development 2007-2013, project number POSDRU/89/1.5/S/56287 ĂPostdoctoral research programs at the forefront of excellence in Information Society technologies and developing products and innovative processesò, partner University of Oradea.

6. REFERENCES

[1] Chen P. M. & Lowell D. E.; (1999). Reliability Hierarchies, *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, pp. 168-173, IEEE Computer Society

[2] Patterson, D. A. & Hennessy J. L. (2006). *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, ISBN 978-0123704900

[3] Stallings, W. (2009). *Computer Organization and Architecture: Designing for Performance*, Prentice Hall International

[4] Howard, P. L. (1990). *The Design Book: Techniques and Solutions for Digital Computer Systems*, Prentice-Hall Inc., Englewood Cliffs, N. J

[5] Avizienis, A.; Laprie, J.-C.; Randell, B.&Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, Vol.1, No.1, pp 11 ï 33, January-March 2004

[6] Huffman, W. &Pless, V. (2003). *Fundamentals of error-correcting codes*, Cambridge University Press, ISBN 9780521782807

[7] Novac, O.; Vari-Kakas, S.; Novac, M.; Indrie, L. & Vladu, E. (2009). Reliabilitty Aspects Regarding the Cache Level of a Memory Hierarchy, *Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium*, 25-28th November 2009, Vienna, Austria, ISSN 1726-9679, ISBN 978-3-901509-70-4, Katalinic, B. (Ed.), pp. 1137-1138, Published by DAAAM International Vienna, Vienna

[8] Zarandi, H. R.&Miremadi, S. G. (2004). A Highly Fault Detectable Cache Architecture for Dependable Computing, *SAFECOMP 2004*, LNCS 3219. M. Heisel et al. (Publisher), pp. 45ï 59

[9] Vari-Kakas, S.; Novac, O.; Poszet, O. & Novac, M. (2005). Reliability Considerations in Memory Hierarchies, *Proceedings of 8th International Conference on Engineering of Modern Electric Systems EMES'05*, 26-28 May, ISSN 1223-2106, pp.149ï152, University of Oradea Publisher, Oradea, Romania

[10] Novac, O.; VlŁduşiu, M.; Vari-Kakas, S.; Hathazi,. F. I. & Novac, M.; (2008). Aspects Regarding the use of SEC-DED Codes to the Cache Level of a Memory Hierarchy, *Proceedings of 7th WSEAS Int. Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED 2008)*, University of Cambridge U.K., 20-22 February, ISBN 978-960-6766-41-1, ISSN 1790-5109, pp. 430-433, Cambridge U.K

[11] Alpert, D. B. & Flynn M. J.; (1988). Performance trade-offs for microprocessor cache memories, *IEEE Micro*, Vol. 8, No. 4, pp. 44ï54

[12] Asadi, G.; Sridharan, V.; Tahoori, M. B. & Kaeli, D. (2005). Reliability Tradeoffs in Design of Cache Memories, *Proc. First Workshop Architectural Reliability (WAR-1)*, in conjunction with the 38th Int'l Symp. Microarchitecture (MICRO-38), Nov. 2005

[13] Vari-Kakas, S. & Novac, O. (2003). Cache modeling and performance comparison by simulation, *Proceedings of 7th International Conference on Engineering of Modern Electric SystemsEMES'03*, Romania 29-31 May, ISSN 1223-2106, pp. 199ï203, University of Oradea Publisher, Oradea, Romania

[14] Novac O.; Gordan, M. & Novac, M. (2005). Data loss rate versus mean time to failure in memory hierarchies, *Advances in Systems, Computing Sciences and Software Engineering, Proceedings of the CISSE'05*, University of Bridgeport, USA, pp. 305-307, Springer

[15] Novac O.; Vladutiu, M.; Vari-Kakas, S.; Novac, M. & Gordan M. (2006). A Comparative Study Regarding a Memory Hierarchy with the CDLR SPEC 2000 Simulator, *Innovations and Information Sciences and Engineering, Proceedings of the CISSE'06*, University of Bridgeport, USA, pp. 369-372, Springer

[16] Novac, O.; Novac,M. &Pакca, V. (2006). Theoretical and experimental study regarding the simulation of a memory hierarchy, *EPE, Buletinul Institutului Politehnic Iaşi,* Tomul LII (LVI), Fascicula E.E.E., pp. 915 ï 920, Iaşi, Romania