Reduction in Cache Memory Power Consumption based on Replacement Quantity

M.Jafarsalehi
Islamic Azad University of Iran
Tabriz-branch
jafarsalehi_m@yahoo.com

M.Reza Salehnamadi
Islamic Azad University of Iran
South-Tehran branch
M_saleh@azad.ac.ir

Abstract

Today power consumption is considered to be one of the important issues. Therefore, its reduction plays a considerable role in developing systems. Previous studies have shown that approximately 50% of total power consumption is used in cache memories. There is a direct relationship between power consumption and replacement quantity made in cache. The less the number of replacements is, the less the power consumption is. In this paper, a mechanism was proposed a mechanism to reduce power consumption using full associative organization and layered replacement algorithm. In this scheme, all cache blocks were divided into three layers. Final layer used FIFO replacement algorithm and middle layer used random replacement algorithm. Also first layer used LRU replacement algorithm. Simulation results were shown using tools written by VB language that in the proposed plan the number of replacement was less than 8-way associative using LRU replacement algorithm.

Keywords: *power consumption, replacement algorithm, replacement quantities.*

1. Introduction

Key limitation power is in the function of processors. In the market of embedded computers in which processors are located in an environment fully relied on disabled cooling or battery power, power consumption is considered as a limitation playing the same role as its functionality and cost. Approximately 50% of total power consumption of processors is consumed in cache memories. Therefore, caches play an important role in power consumption generation taking its reduction as an important issue in developing modern systems. (Malik,2004) Power consumption includes static power resulted from leakage current and dynamic power from logical switching current, charge and discharge of load capacitor. Currently, several methods have been proposed to reduce power consumption totally divided into four various groups.(Abella, 2006)

1-1: Pseudo-Associative Caches

Many plans have been offered that they have been trying that cache with miss ratio equivalent to associative cache, delay and power dissipation equivalent to direct mapping. Some of these plans are concentrating on the access time reduction (Batson, 2001) and others are based on predicting the way where the data are stored. (Powell, 2000).

1-2: Non-Resizing Low Power Schemes

In order for decreasing power consumption, many plans have been proposed through change in cache organization such as placing a cache as filter before L1 cache (Kin, 1997), applying certain modules for variety of accessibilities, partitioning and various coding for cache memories and varied techniques for dividing and isolating bit lines and etc. (Abella, 2006)

1-3: Resizing Low Power Schemes

Power consumption reduction can be achieved through re-configuration statically or dynamically by some characteristics such as size, associativity and active ways and/or through deactivating memory lines. Cell content is lost by connecting SRAM memory cell feed supply in which its content is not required to gate or ground route, but there would be no leakage current at all.(Powell, 2000), (Agarwal, 2002) Also it is possible to reduce leakage current without contents being lost by decreasing feed supply in a number of cache lines and placing them in drowsy mode (sort of sleep mode). (Alioto, 2005)

1-4: Low Miss Rate Schemes

Several approches to reducing the miss rate and/or complexity of conventional caches have been proposed conventional caches use of a subset of bits from the address to index the tag and data arrays, as dictated by the modulo function.(Abella, 2006) Some authors have shown that other function provide lower miss rates since they reduce the number of conflicts.(Karbutli, 2004). Other approaches have focused on the replacement function to reduce the cache miss rate. Different replacement functions have been proposed to improve the performance of LRU for L_2 caches (Wong, 2000) and to enable the compiler to provide hints to the cache system for replacement of those cache lines that are not likely to be reused soon(Wang, 2002). A non-uniform cache architecture (NUCA) for wired-delay dominated is proposed in (Kim,2002) on-chip caches and a scheme to place the data in these kind of caches to obtain low miss rates and reduce latency. An improved version of NUCA is presented in (Chishti,2003)

2. Replacement Policy

Designing a cache system consists of several steps including cache size, block size, association degree and etc. One of the significant decision in designing is to select appropriate replacement policy. Replacement policy impacts on the general system efficiency.

There are two aspects while applying replacement algorithms in the embedded processors. The first one is miss ratio. A miss ratio leads to performance penalty and more energy consumption. (Cho, 2009) has shown that energy consumption in case of miss ratio can be tripled for consumption in hit time. Also when the processor is in idle mode, it is wasted. Therefore, selection of a replacing method leading to less miss not only reduces execution time but also causes to optimize energy return. The second aspect is that implementing replacement algorithm should be easy and efficient.

Many other algorithms have been offered to decrease miss ratio except classic algorithms such as FIFO, random and LRU. J. O'Neil (1993) offered LRU-K algorithm.

In LRU-K, k^{th} backward distance is stored dynamically that is defined as the number of referrals within a period of time from last k^{th} referral of the respective block to its most recent referral. A block with maximum k^{th} backward distance is chosen in case of occurring miss ratio for replacement. (Johnson ,1994) offered 2Q algorithm as optimized from LRU-2. This algorithm has two rows as A1 and Am. In the first referral to a 2Q page puts in row A1. If a time page located in rows A1 and Am is referred again, that page is rejected to the end of row as Am. Size of A1 and Am is fixed (e.g. for A1 20% of cache size and for Am, 80% of cache size). When a new page is added to one of the rows, the old page should be omitted if necessary. (Megiddo, 2004) proposed LIRS algorithm in which Inter Reference Recency (IRR) is used as historical information of each block. IRR in each block is regarded as the number of other blocks locating between two permanent accesses to the respective block. Whenever a block is chosen for replacement, IRR blocks data are utilized. This algorithm dynamically distinguishes the blocks with less IRR from blocks with high IRR. Blocks with less IRR are maintained in cache. Blocks recency is only used to determine the status of LIR or HIR blocks. (Lee, 2001) offered an algorithm, combined of LRU and LFU called LRFU. This is a subset of LRU and LFU. Each page X is attributed with c(x) =0. It is updated after each access to cache depending on parameter $\lambda > 0$. If block x is referred, we have $c(x) = 1 + 2^{-\lambda}c(x)$, otherwise, it will be $c(x) = 2^{-\lambda}c(x)$. LRFU replaces a page with the least amount of c(x). Performance depends on λ. (Megiddo ,2004) proposed ARC algorithm. It has page lists of LRU as L1 and L2. L1 stores those pages recently observed only once and L2 maintains the pages recently seen at least twice. This algorithm only caches the deduction from pages available in the lists. The pages seen twice in a short time are more repeated or they are potentially reusable for a longer period of time. If the cache can store C page, so we should have two lists in the same size of C. ARC can cache a variable number of pages recently used L1 and L2, so that the total number of cached pages will be C. ARC permanently adjusts a certain number of pages from each list previously cached.

3. New proposed scheme

In a full associative organization, each block can be located every place in the cache, so enjoying a higher hit rate compared to direct mapping and set associative. But a full associative has some drawbacks including while cache is full and a new block is due to enter, it is necessary to remove one of cache blocks. It is difficult to use LRU replacement algorithm to implement it and even impossible when the size of cache is large.

In the proposed method, a full associative and order residence plan was used, and all cache blocks were divided into three layers in order to decrease the complexities of full associative: first layer, middle layer and last layer. A specific replacement method was used in each layer. If we assume the number of cache blocks as N, there are 5N/8 blocks in the last layer, N/4 in the middle layer and N/8 block in the first layer. Assuming, there are four blocks for N=32 in the first layer, eight blocks in the second layer and twenty blocks in the last layer. The new block will always be located in the last layer and the block which was assumed to be removed from cache was located in the first layer. The replacement method was LRU in the first layer. The replacement method was chosen randomly in the middle layer and FIFO was selected for the last layer. Whenever it is necessary to remove a block from cache, the replacing block was determined by

LRU algorithm in the first layer and then was removed. Then a block from the last layer was located in the middle layer determined by FIFO algorithm. Accordingly, a block was emptied in the last layer and a new input block was located in the middle layer. Since, it is difficult to implement LRU algorithm, and also it is more efficient compared to other classic algorithms, this algorithm was only used for the first layer in which the number of blocks is least compared to two other layers. FIFO and random algorithms are easier, that's why these two algorithms were used in the middle and last layers.

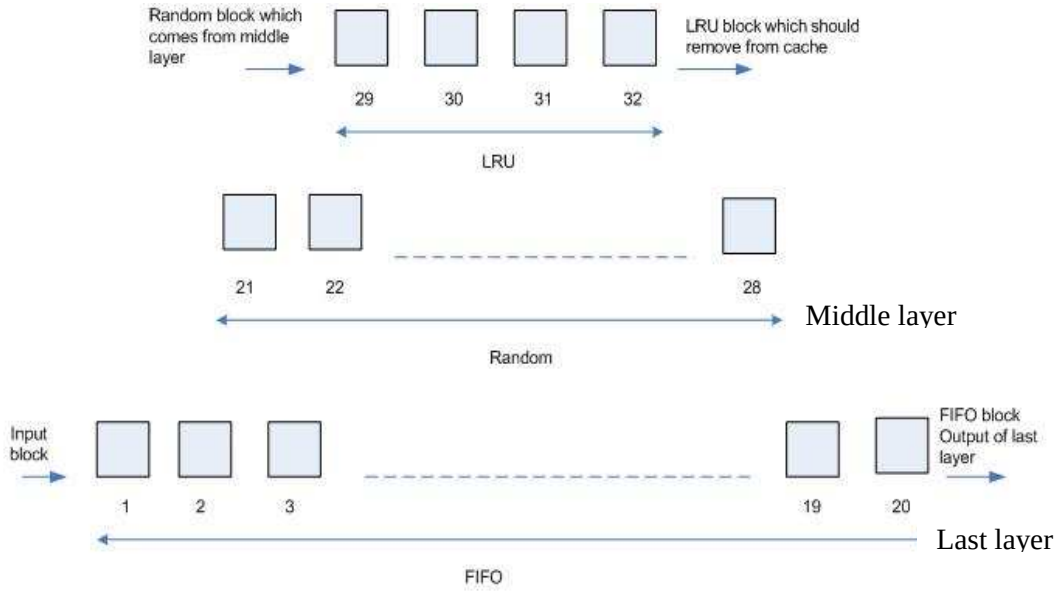


Figure 1: *The proposed plan when there are 32 cache blocks.*

In a full associative organization, all cache blocks should be checked to find an empty block that it takes too long and also it needs a First layer l hardware for implementation. Therefore, it leads to reduction both in runtime and performance, whereas in the proposed plan, the location of empty block was always specified and new request was located in first block of last layer.

3-1: LRU implementation in proposed scheme

(Grossman ,2002) has implemented LRU replacement algorithm using systolic array. In this technique, a list of indices for all cache lines is maintained regularly from LRU to MRU. When a cache line is accessed, its line index is appeared in the list and that index is rotated toward MRU position in the end of list. Figure2 depicts the LRU list. This list can be implemented by systolic array. There is a bit called matched bit showing if there is an index for L line in the array or not. In any clock cycle, while matched bit is zero, L or line index is moved forward without any change, but when matched bit became one, node content is copied from right side, which is shown figure3.

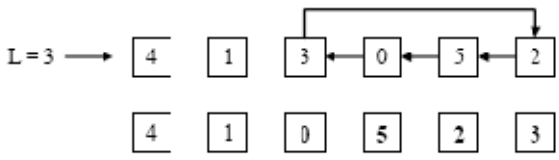


Figure2: *LRU List*

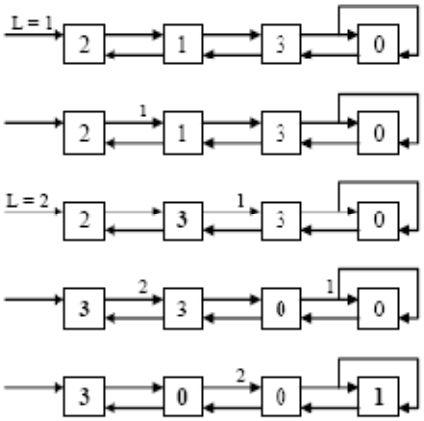


Figure3: *Systolic Array implementation*

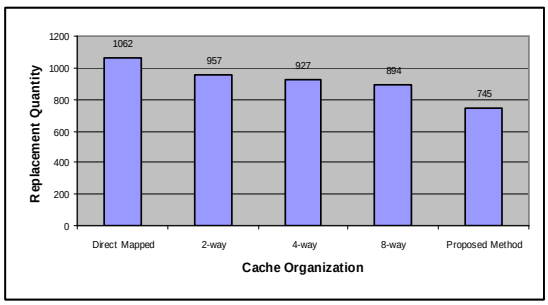
The above method was used in the proposed scheme in order to implement LRU algorithm. In this case, the position of replacement block was determined and it does not take too long to find replacement block unlike full associative organization.

4. Simulation

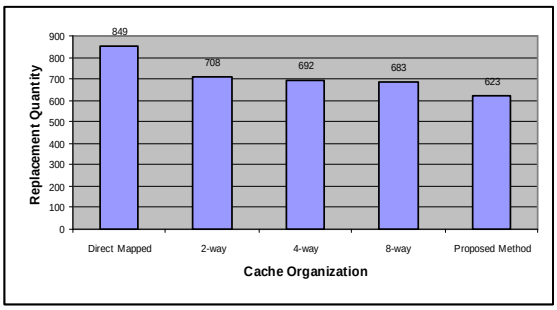
Simulation was performed using software written in VB language. In this simulator, some parameters including size, organization and cache replacement algorithm can be arbitrarily adjusted. Simulator worked in a way that first a certain request quantity was randomly created, then each parameter related to cache was adjusted and finally the result of simulation as replacement number was shown.

4-1: Simulation Results

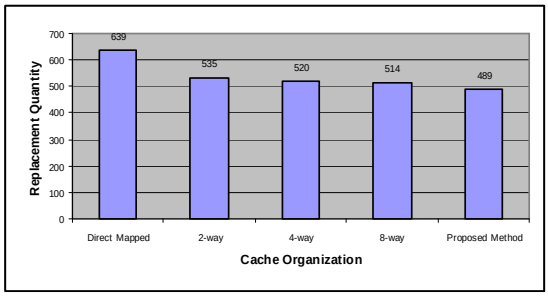
Figure 4 shows the results of simulation while the number of requests was roughly ٢٢00 and the number of main memory blocks equals to 2048. LRU replacement policy was used for set associative organization in simulation.



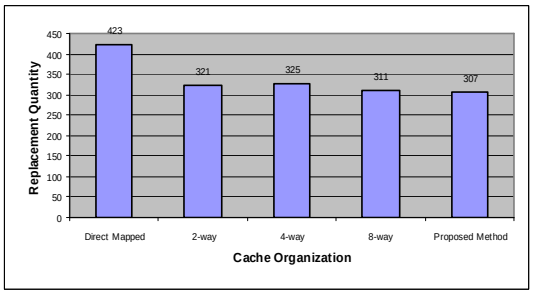
a) The number of cache blocks equals to32



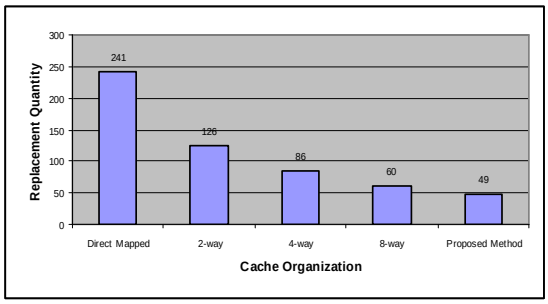
b) The number of cache blocks equals to 64



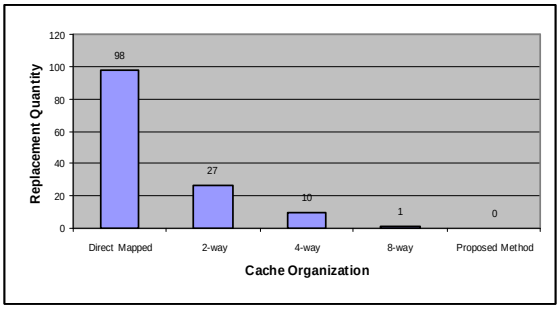
c) The number of cache blocks equals to128



d) The number of cache blocks equals to256



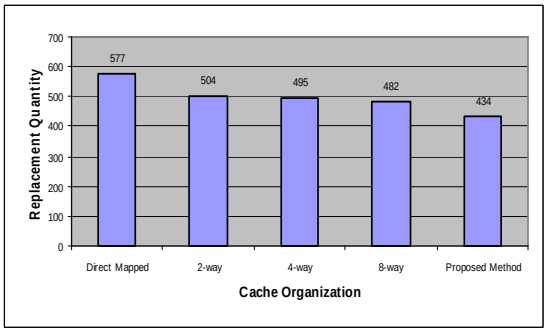
d) The number of cache blocks equals to 512



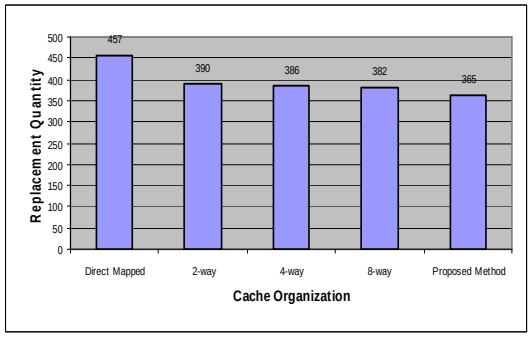
e) The number of cache blocks equals to 1024

Figure 4: *2200 requests and 2048 main memory blocks*

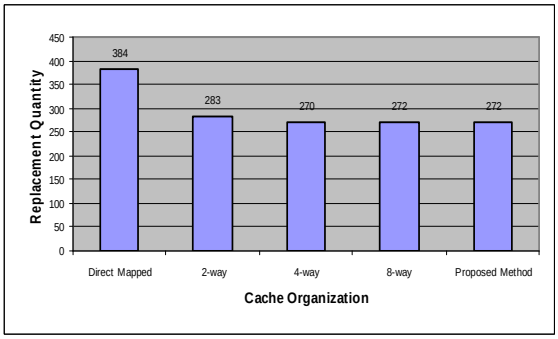
Figure 5 shows the results of simulation while the number of requests approximates to ۲۲00 and the number of main memory blocks equals to 2048. LRU replacement policy has been used for set associative organization in simulation.



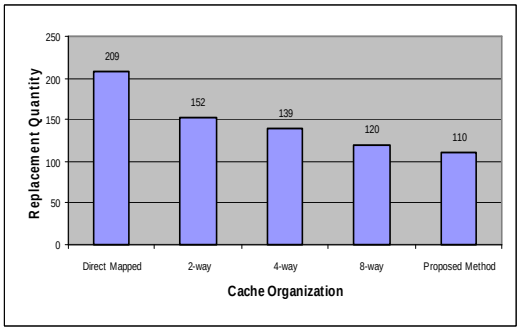
a) The number of cache blocks equals to32



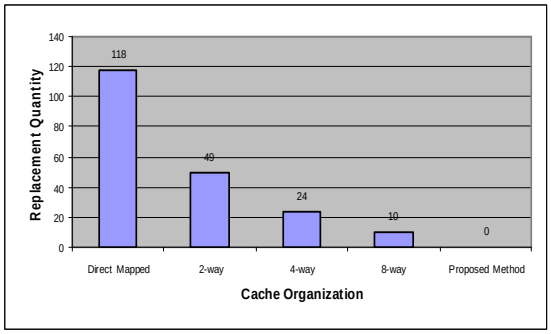
b) The number of cache blocks equals to 64



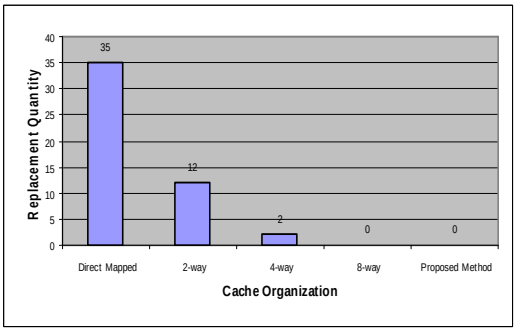
c) The number of cache blocks equals to128



d) The number of cache blocks equals to256



d) The number of cache blocks equals to 512



e) The number of cache blocks equals to 1024

Figure 5: *1200 requests and 2048 main memory blocks*

The findings based on figure 4 and figure 5 are:

1. *Direct mapped organization had maximum replacement between other organizations.*
2. *Increase in associativity led to less replacement.*
3. *Replacement quantity in proposed method was less than replacement quantity in 8-way associative organization.*
4. *Cache blocks increase in all organizations led to less replacement.*
5. *Proposed method had less replacement among other cache organizations.*
6. *Use of various replacement algorithm in set associative organization had no influence over 1,2,3,4,5 results.*

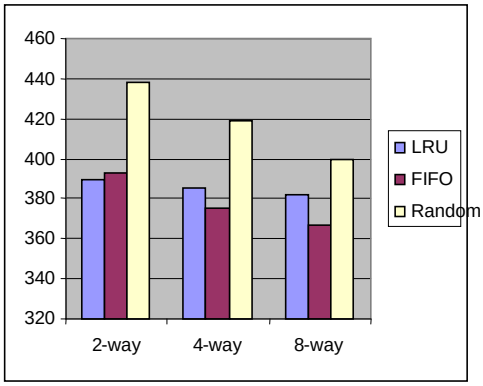
Table 1 shows the number of replacement in various cache sets with varied requests.

Table 1: *The number of replacement in various cache organization in different main memory sizes with the number of varied requests and at least 256 cache blocks*

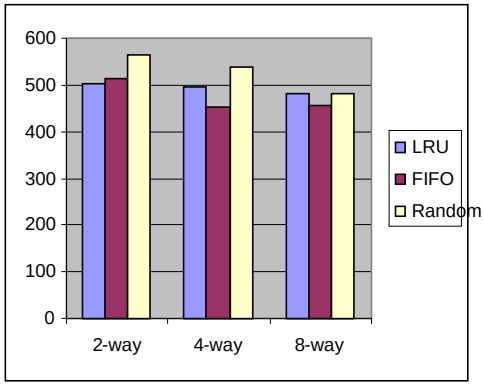
The number of cache blocks	The number of main memory blocks	The number of requests	The number of replacement			
			Direct Mapping	2-way	4-way	Proposed Plan
256	4096	3500	1295	1790	0	0
256	8192	3500	1542	1367	802	421
256	32768	3500	1293	1718	1624	275
256	4096	4500	1420	1907	676	649
256	8192	4500	2193	2192	1018	925
256	32768	4500	1674	2234	2096	344
512	4096	5500	171	73	73	0
512	8192	5500	118	85	34	0
512	32768	5500	218	46	46	0

Table1 shows that main memory increase or requests number had no influence over the obtained results and the number of replacement in the proposed scheme was minimum compared to other cache organizations obtained in the table. Also, simulation was performed using FIFO and Random replacement algorithms and same results obtained.

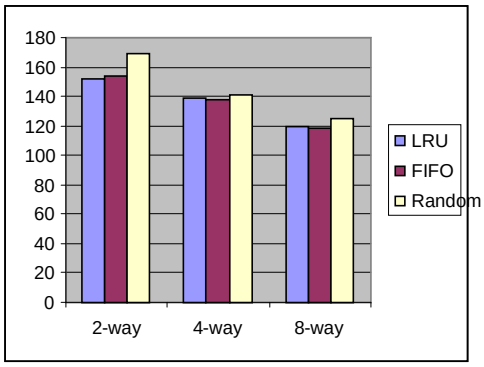
Figure 5 shows replacement quantities in set associative organization while number of requests equal to 1200 and number of main memory blocks equal to 2048.



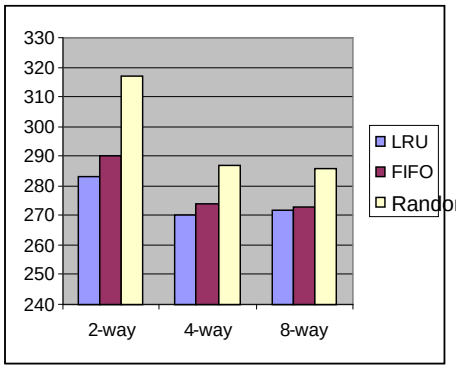
b) The number of cache blocks equal to 64



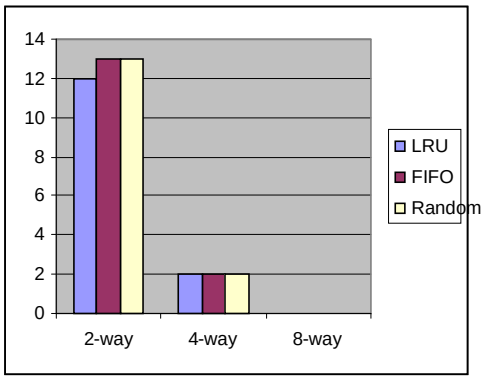
a) The number of cache blocks equal to32



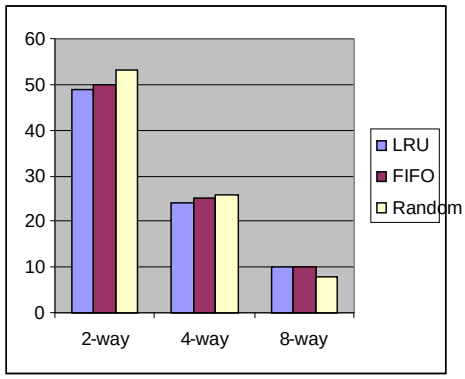
d) The number of cache blocks equal to 256



c) The number of cache blocks equal to128



f) The number of cache blocks equal to 1024



e) The number of cache blocks equal to512

figure 5: *Replacement quantities in set associative organizations using different replacement algorithm*

Figure 5 shows LRU replacement algorithm had less replacement quantities compared to FIFO and random policy. However FIFO in some cases had a better hit rate than LRU which could be neglected. Random policy in all cases causes more replacement compared to FIFO and LRU. For these reasons and according to hardware implementation, in proposed scheme, the first layer used LRU policy, middle layer used random policy and first layer used FIFO policy for replacement.

Conclusion

In this paper, a new scheme of a full associative was presented. The new scheme was easier to implement compared to a full associative main scheme. Results of simulation were shown using a cache simulator written in VB language. The number of replacements in the proposed scheme was less than in 8-way associative and since power consumption is directly connected to the number of replacements, this scheme could also cause reduction in power consumption. The proposed scheme was flexible and other replacement algorithms with low miss rate could be used in each of three layers.

References:

1. Abella, J., 2006,Hetergeneous Way-size Cache, Proceedings of the 20th annual international conference on Supercomputing, Cairns, Queensland, Australia ,SESSION: Memory, pp. 239 - 248
2. Agarwal, A., Li,K., Roy, K., 2002, DRG-Cache: A Data Reduction gated-ground Cache for Low Power . In DAC 2002
3. Alioto, M., Bartolini,R., Bennati, P., Giorgi, R., 2005, New Techniques for Low Power Caches, ACACES 2005 – Poster abstracts 133
4. Batson, B.,vijaykumar, T. N,2001, Reactive Assiciative caches. In PACT 2001
5. Cho. S.,2009,Augmented FIFO Cache Replacement Policies for Low Power Embedded Processors, Journal of Circuits, Systems, and Computers, vol.18,no6,pp 1081-1092
6. 3-Grossman,J., 2002, A Systolic Array for Implementing LRU Replacement
7. J.O'Neil, E.,1993,The LRU-K Page Replacement Algorithm for Data-Base Disk Buffering, ACM SIGMOD record,vol22,Issue 2,pp.297-306.
8. Johnson, T., Shasha, D., 1994, 2Q: A Low Overhead High-Performance Buffer management Replacement Algorithm, Proc VLDB Conf., Morgan Kaufman, pp. 297-306
9. Kharbutli, M., Irwin, K., solihin, Y., Lee, J.,2004, Using Prime Numbers for cache Indexing to eliminate conflict Misses. In HPCA 2004
10. Kim, C., Burger, D., Keckler, S.W., 2002, An Adaptive Non-Uniform Cache Structure for Wire – Delay Dominated On-chip Cachees. In ASPLOS 2002
11. Kin, J., Gupta, M., Mangione-Smith, W.H., 1997, The Filter Cache: An Energy Efficient structure . In MICRO 1997
12. Lee, D., 2001,LRFU:A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies, IEEE Trans. Computers, vol 50,no 12, pp.1352-1360.
13. Malik,B., 2000, A Low Power Unified Cache Architecture providing Power and performance Flexibility, Int. Symp. On Low Power Electronics and design, June 2000
14. Megiddo,N., Modha, S., 2004, Outperforming LRU with an Adaptive Replacement Cache Algorithm,IEEE,vol 37 ,pp.58-65
15. Powel,M., Agarwal, A., Falsafi, B., Vijaykumar, T.N., Roy, K., Reducing Set-associative Cache Energy via Way-Prediction and Selective Direct Mapping, In ISLPED 2000
16. Wang, Z., McKinley, K.S., Rosenberg, A.L., Weems, C.C., 2002, Using the Compiler to Improve cache Replacement Decisions. In PACT 2002
17. Wong W.A., Bear, J-L., 2000, Modified LRU Policies for Improving Second-Level cache Behavior. In HPCA 2000