Performances evaluation of multi-modules caches memories for embedded systems

Hajer.CHTIOUI[1], Smail.NIAR[2], Mohamed.ABID[1]
[1]CES Laboratory, University of Sfax Tunisia
[2] LAMIH, University of Valenciennes, French
chtioui_hajer@yahoo.fr, smail.niar@univ-valenciennes.fr, mohamed.abid@enis.rnu.tn

***Abstract*—The effectiveness of caches memories in term of energy consumption and execution time represents an important challenge in the embedded systems design. In response to these needs, new structures of caches appeared. This paper concerns these structures and their effectiveness to improve embedded system performances. In this context, we modelled and evaluated the technique of multi-modules caches "cache of the victims" with the simulator of embedded architecture SimpleScalar/ARM. The results obtained are interesting. Indeed, the rate of improvement of the miss ratio arrives on average up to 53%. The energy consumption is quite reduced for all the benchmarks this is noted especially for the benchmark (cjpeg) which has a reduction ratio of the energy consumption which arrives up to 42%.**

Keywords-component; Multi-modules caches memories; embedded systems; victim cache Simplescalar/ARM; energy consumption

I. INTRODUCTION

With the appearance of the modern applications such as the multimedia (audio, video, plays…) and telecommunication, the embedded systems become more complex. One of the serious problems with these applications is that they require increasingly an important flow of data. Therefore, they need an intensive use of the memory. However, the consumption and the time access of the memorizing units largely dominate the overall consumption and the total performances of such applications. Consequently, the respect of the constraints of reliability, time design, surface, consumption and guarantee of the performances becomes very critical. So, the memorizing units play a very significant role in the embedded systems. For this reason, the designers always deal with the improvement of the memories and more precisely the cache memories.

On the one hand, the cache memory becomes a basic and essential mechanism in reducing the latency of access to the memory and the consumption of energy. On the other hand, the increase of miss ratio in caches involves a slower path of data, which leads to a slower time of cycle and much more energy consumption, consequently reduction of the performances of the global system. The basic idea of the classic solutions [1] aims to exploit the characteristics of the cache memory either by increasing its size or by improving its degree of associativity. However these alternatives give reduction in the speed of research of the block in case of success. It is important to note that the size factor is not always exploitable for the miniaturized systems which require the surface constraint. Then the embedded systems designers have to propose new structures of cache which must face those problems. A variety of structures of caches appeared as a response to those needs. In this paper we focus to study the multi-models structures [2] [3] [4] [5]. They especially deal with the improvement of the data caches in the first level (noted L1 Dcache) seen their importance essentially in signal processing applications which use an enormous quantity of data.

The idea based on adding the L1 Dcache a small module of cache and it is possible to be extended, that will be named the multi-modules. This module is called assistance cache or still auxiliary cache. The main purpose is to increase the chance to have a cache success by exploiting the principles of temporal and space locality. The temporal locality is exploited by charging the data in L1 Dcache at time of the first reference (first miss), whereas the space locality is exploited by charging not only the referred data but also the data neighbouring (a data block).

Within this framework, the objective of this paper is to evaluate the performances of these structures in terms of time execution and energy consumption. The stress is laid on their utility in the SoCs (System on Chip) for the applications of intensive signal processing. Work aims to implement on the simulator of embarked architecture SimpleScalar/ARM [7] a technique of multi-modules caches. This technique [2] is named cache of the victims (noted VC). This paper consists of three sections. The first section is a representation of the principal works to carry out on the multi-modules caches memories. In the second section, we detail the key points related to the design of model VC. The third one, initially represents the environment of integration and simulation which we adopt. It gathers thereafter the various results from integration.

II. STATE OF ART

A variety of research works is realized on the multi-module caches of which we represent some of it in this part. These structures have the same principle that is the addition of an auxiliary cache to L1 Dcache (figure 1). But they are differed by the way in which they are disposed. The techniques of exchange between them and most interesting are their manners with which they exploit the

principle of locality. We classify these structures in two categories. The first category determines the locality of the block by seeing its behaviour at the time of the current turning the other word their tendency to be referred another time. The second determines the locality of the block by seeing its history at the time of the turns which passed previously.

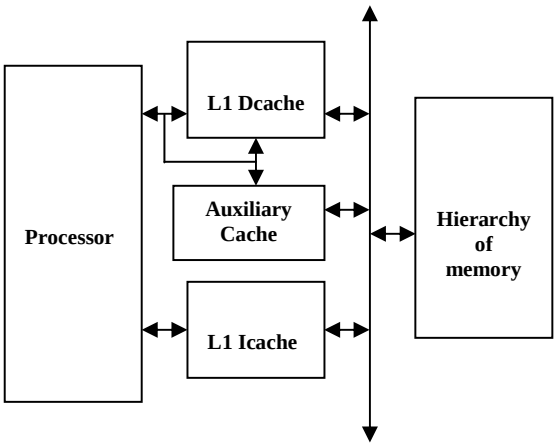

Figure 1. Principal of Multi-modules caches memories

A. *Multi-modules cache Memory out of background*

1) *Caches of the victims*

It is agreed that the direct mapped cache (noted DMC) is simple and inexpensive to implement. But its main disadvantage is the fixed site of the block in cache. Consequently, if a reference program in a repetitive way words of two different blocks which correspond to the same line, these blocks permute continuously in cache, and the success rate becomes weak. The traditional solution of this problem is to increase the associativity of cache, yet this will require an increase in the access time to cache since the research of the block will become more complicated. The standard of researcher Jouppi [2] observed that in the majority of the cases, only few, blocks of cache which are responsible for a defection in cache. Therefore, it proposed to improve L1 Dcache by adding a fully associative cache to it (noted FA) named cache of the victims. The idea is to put the ejected blocks of L1 Dcache in VC hoping that the next time there needs this block it can be consulted quickly from the VC instead of going to seek in the level 2 cache memory (noted L2 Dcache). The new cache is named cache of the victims because we find in it only blocks ejected in case cache defection.

2) *ABC « Allocation By Conflit »*

This structure [3] comes for the improvement of DMC cache by adding a cache FA to it. It is similar to structure VC but this case the decision to put a block in the principal cache called A (DMC) or in the secondary cache called B (FA). This fact is considered the turn running of the block in conflict. Each time that an entering block which is put in B. We say that a CNR (conflict without replacement) occurred in A. With each block in A is associated a bit C that indicates if a CNR is produced in A since its last reference. Bit C is 0 in its initial positions. Each time a CNR occurs, C is put at 1 for all blocks of A. the bit C of a block is given to zero each time the block is referred. SoC is equal to 1 indicates that at least a CNR is produced in A during the current turn of this block which was not put in reference from since last CNR. The block of conflict is considered actively referred if its bit C is 0.

B. *Multi-modules cache memory referring to its background*

1) *Dual Cache*

This technique [4] is composed of two caches S and T disposed in parallel and a mechanism of prediction (noted LPT for Locality Prediction Table). The LPT contains the history of the instructions memories most recently performed. The blocks in T are smaller than those in S. The LPT indicates not only the type of locality associated with the instruction memory which caused the loading of the block (S, T or no locality), but it contains also information allowing a preloading of the data. The mechanism of prediction is used only when there is a miss. In other words, a data T (having a large temporal locality) can lie very well during a turn in the memory S, because it is in the same block as a data S. Prediction on the level of the LPT is made by taking of account the address instruction, the difference in the reached addresses (the step), and the length of the vector. A data can be localized in the two caches.

Indeed as the blocks have different sizes, the data can belong to a block where there has a data S and is another block (with another size) or there is a data T. In this case, to avoid problems of inconsistency, the reading is done in priority in T and when a block of T is purged, it updated there data in S.

2) *NTS « No Temporel Streaming »*

The goal of this structure [5] is to increase the chance to have a success of cache. This is doing by safeguarding longest possible in L1 Dcache the blocks with large temporal locality. The other blocks which do not have a good temporal locality are recovered in an auxiliary mask added to L1 Dcache. The decision to put the block entering the principal hiding place or that auxiliary is done by testing the history of the block in the turns precedent and this is using a table of history (noted for Detection Links) which keeps information on the blocks which forward by L1 Dcache.

III. STRUCTURE TO BE INTEGRATED

A. *Structure VC*

A L1 Dcache that using a VC is like that which is already represented by figure 2: a large cache DMC and a small auxiliary cache FA (1 KB). This latter is laid out behind the DMC, and plays the role of interface with the L2 Dcache. The search for a block is done in parallel in the two caches. It is then enough to have a success in the DMC. The VC will not reach and the sought word will be delivered to the processor. In case of miss in the DMC, and that there is a success in VC cache then the block victim which will be ejected DMC and the desired block of VC cache are permuted. In case of miss in the two caches, the

block is read from the L2 Dcache then safeguarded in the DMC.

The block which will be replaced is sent to the VC for a future use. This small cache is laid out behind the principal cache consequently it does not have a direct way between VC and the processor. But there is a bidirectional way between the DMC and the VC for that the data path, is expensive especially when there is a success in the VC.

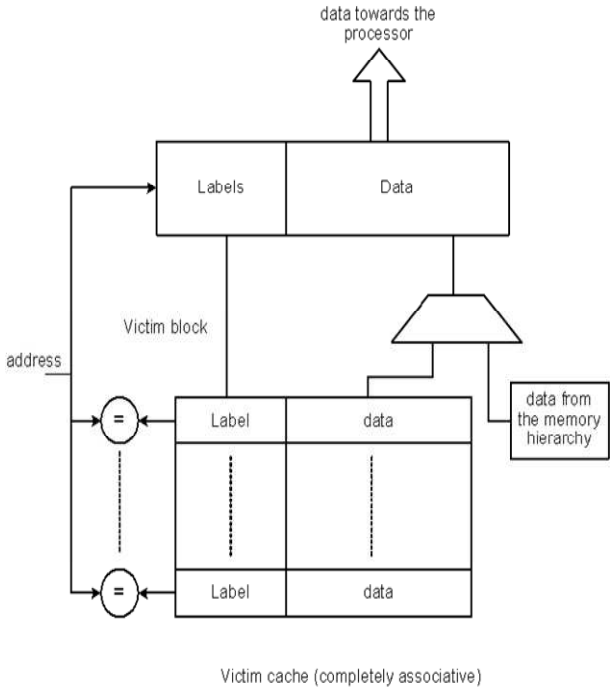

Figure 2. Structure of the L1 Dcache with the cache of the victims

B. *Integration and simulation environment*

In this part we present the environment adopted for our experimentation and we detail our design. We choose punt forms simulation SimpleScalar/ARM version 3.0 [7], to simulate and to evaluate the VC model. Indeed, this punt form has a whole of very rich tools with a good execution during simulation. Its code written with C language, simple and modular allowing the user to add and modify any architecture.

1) Implementation

In this section we indicate the stages which we preceded in the implementation of structure VC.

Among the variety of the modules that obtain SimpleScalar/ARM. we are interested in the modules of cache "cache" and "sim-cache". The implementation module "cache" is characterized by a generic controller of a memory cache at the level of view size and associativity. The reading in case of miss is "Read-Through" i.e. the reading is directly completed from main memory to the CPU. The writing is "Write-back" which means that the block is written only in the memory cache. We keep these characteristics for the structure of cache which we modeled.

The module "sim-cache" generates the statistics of the memory cache and the configuration TLB for "Translation Look-aside Buffer" (structure of cache which safeguards the translations of the virtual addresses which attain the cache in physical addresses). Their characteristics size and associativity are fixed by the user. It simulates up to two levels of instruction and data caches (they can be unified), with an instruction and data TLB level.

Each time that there is an access to a memory cache the module "sim-cache" calls the functions "cache_create" and "cache_access" of the module "cache". The function "cache_create" realizes the creation of a memory cache whose definition is carried using the structure "struct cache_t". The function "cache_access" reaches the cache in writing or reading according to the order to be executed.

The quite detailed study of the principal functions related to the design and the simulation of a memory cache according to SimpleScalar, enables us to proceed as follows:

We implement in the module "cache" two new functions. The first function named "cache_access_new" which gives access to the principal cache DMC of technique VC. While the second function "cache_access_vc" gives access to the secondary cache VC of this technique.

2) Simulation results

We test the module "sim-cache" before the integration of our model VC with six benchmarks of the MiBench suite (bf, rijndeal, PGP, fft, sha, cjpg) for a size of L1 Dcache of 4KB and whose technique of placement is DMC. Then, we realize the same tests for our module "sim-cache" after the integration of model VC with size 1 KB and whose technique of placement is FA.

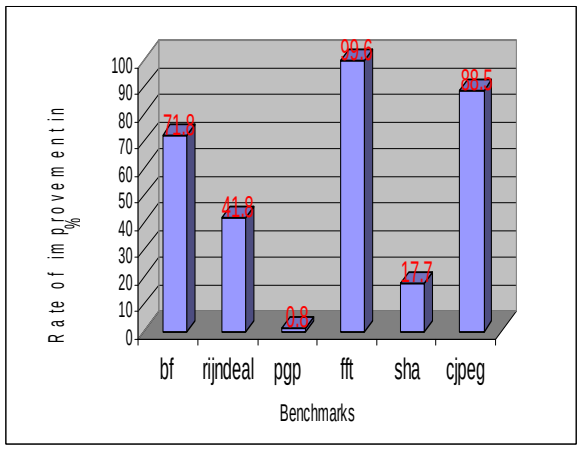

Figure 3. Improvement rate of the miss ratio for a L1 Dcache with size 4KB

Figure 3 shows the improvement rate of the miss ratio according to different benchmarks for a size from cache of 4KB. We note that the rate of improvement of the miss ratio is very significant for the benchmarks bf, rijndeal, fft and cjpeg. Indeed, this rate attains on average up to 53% what reflects the effectiveness of the integration of our technique.

The reduction in miss ratio involves a reduction in the total number of access at the lower levels of the hierarchy

memory. We affirm this by figure 4 which represents the improvement rate of total number of access to the L2 Dcache according to the benchmarks.

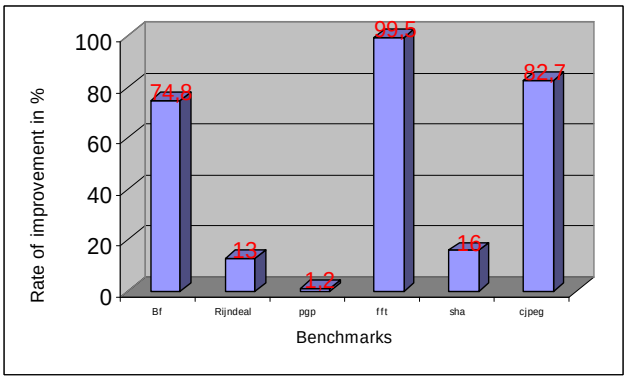

Figure 4: Improvement rate of the access total number to the L2 Dcache for L1 Dcache of 4KB

In order to evaluate the energy consumption of the multi-modules cache structure VC, we used the energy estimate model of CACTI 3.0 [6]. We recovered the results illustrated by figure 5.

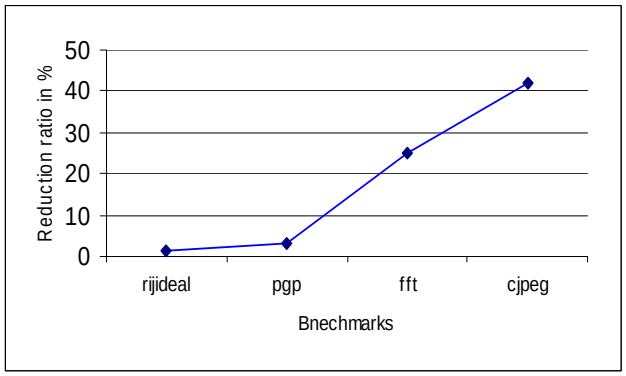

Figure 5: Reduction ratio of the energy consumption according to the benchmarks

We prove referring to these results that the integration of technique VC in SimpleScalar/ARM is very efficient for the improvement of its performances.

We note that the energy consumption of the memory cache is quite reduced with the integration of the multi-modules cache structure VC. Indeed, this is marked especially for the (cjpeg) benchmark which has a reduction ratio of the energy consumption which arrives up to 42%.

IV. CONCLUSION

The cache memories considerably drew the attention of the designers as of their appearance. They are particularly attractive for the embedded systems which handle applications of intensive signal processing.

Indeed, this memory is the best response to the needs for these systems in term of the energy consumption and time response. Consequently, the designers are motivated in order to improve the performances of the cache memory. This motivation is well marked by the appearance of several new structures of caches such as the multi-modules structures.

In this paper we modelled, simulated and evaluated the performances of the multi-models cache structure VC. The results of simulation which we obtained have makes it possible to validate the correct function of our application. In addition, they made it possible to evaluate the performances of model VC in term of energy consumption. Indeed, we referring to the results of simulation that the integration of this model in SimpleScalar/ARM improves the L1 Dcache miss ratio of a value which arrives on average up to 53%. Consequently, we obtain at the level of the energy consumption. Thus the reduction of the energy consumption arrives up to 42% with the benchmark (cjpeg) of the MiBench suite. In conclusion, model VC made a success of the L1 Dcache improvement thus the improvement of the total system.

Currently, the tendency is oriented to the design of multiprocessor embedded architectures (MPSoC). So, we minks studied the effectiveness of the multi-modules caches memories for these systems.

REFERENCES

[1]http://www.pearsoneducation.fr/Documents/Catalogues/stallingschap4.

[2] N.P. Jouppi, “Improving Direct-Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers,” Proceedings of ISCA-17, pp. 364–373, June 1990.

[3] E. S. Tam, “Improving Cache Performance Via Active Management,” Ph.D. Dissertation, Dept. of EECS, University of Michigan, Ann Arbor, 1999.

[4] A. Gonzalez, C. Aliagas, and M. Valero, “Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality,” Proc. Int’l Conf. on Supercomputing’95, July 1995, pp. 338-347

[5] J. A. Rivers and E. S. Davidson, “Reducing Conflicts in Direct-Mapped Caches with a Temporality-Based Design,” Proceedings of the ICPP, vol. I., pp. 151 - 160, August 1996.

[6] P. Shivakumar and N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model.

WRL Research Report 2001/2, Aug. 2001.

[7] http://www.simplescalar.com