Heuristic Prediction for L1 Data Cache Misses for Low Power Cache Memory to Reduce the Cache Miss Penalty

Pardeep Malik
Assistant Professor
Department of Computer Engineering
G.L.Bajaj Institute of technology & Management, Greater Noida Utter Pradesh

Abstract— Ever since a long time ago, memory accesses costs about half of a microprocessor system's power consumption. Modifying a microprocessor cache's total size, line size and associativity to a particular program request have tremendous benefits for performance and power. Customizing caches has until recently been restricted to core based flows however, several configurable cache architectures have been proposed recently for use in pre-fabricated microprocessor platforms. On chip-level techniques alone cannot further keep power dissipation under a reasonable and acceptable level. We show in this paper that many researchers have already been made efforts to reduce power dissipation at the architectural level by reducing on-chip cache power consumption which is a major power consumer in microprocessors. In this paper we propose a runtime mechanism that heuristically predicts the throughput of an application using a reconfigurable low power L1 cache. In this paper we propose on-chip hardware implementing an efficient L1 cache tuning heuristic prediction that can automatically, transparently, and dynamically reduce the cache miss penalty to an executing program. There are other approaches that follow the program flow and prefetch all target addresses in L1 data cahe including those blocks which already exist in the L1 data cache. The approach predicts the stream of next miss heuristically using genetic algorithm according to recency of data accessed by the user and then prefetches only the next miss address of the stream. It uses a general prefetching framework, two-phase prediction algorithm (TPP), that lets each stream request have its own address predictor comparing the TPP algorithm with the latest variant algorithms of stream buffers and Markov predictor. In fact, the cache miss penalty can vary almost one or two orders of magnitude because of a huge divergence between sequential and random access rate of disk. Our heuristic seeks not only to reduce the number of cache misses that must be examined, but also traverses the search space in a way that completely avoids costly cache flushes. Goal of this paper is to present the solution for prediction information in low power L1 cache in order to use heuristically the cache operation with usage Genetic algorithms.

Key words: Heuristic Prediction, Low Power Cache Memory

I. INTRODUCTION

Studies till now shows that most of the power consumption in current processors come from the cache memory [1,2]. This fact has gained more importance today only because there is a high demand on mobile devices that need low power design. Some of the few common high-level energy efficient techniques that have been studied by different papers have block buffering and cache sub-banking. Due to this high demand of the industry, the challenge in keeping up with Moore's Law is been getting harder. In line with this, the new trend in the engineering community is to put more processors in a single chip. However, building a multiprocessor system produces different issues which are not seen in a single processor. Some of these are the memory coherency for each cache connected to each processor & simultaneous accesses to a single shared memory block.

Thanks all to these technology users can compute even through walking mobile user desire to access privately or share Databases which saved on mobile or immobile hosts and offer the investigation and up to date the data in wire and wireless webs. There are few disadvantages in this kind of service such as limitation in nurture sources for mobile networks high expenses & limitation of band-width in mobile networks which causes disconnection b/w mobile network and the mobile service station (MSS). The disconnection re-dounds transaction abort and mobile systems have to redo transaction from start. This will need extra expenses such as connection to network and additional time. One method of preventing transactions at the mobile service station as receives the mobile networks request predicts the additional data & sends them to the mobile network. The mobile networks save the additional data in its cache memory and in the case or emerging appealing data. In local memory through application of the transaction on next ones makes use of them so there is not any need for connecting and sending request for data .this prevents transaction abort due to disconnection therefore it is helpful in reducing the expenses and improving responding to the user of mobile system. The most important factor in this technique is exact prediction of data for sending which needs data excavating in database. We will use genetic algorithm to do this.

II. BACKGROUND

Data cache misses form an important bottleneck of the contemporary processors. Among these proposed solutions, prefetching schemes will preferred, as they have the potential to eliminate not only conflict misses but also capacity and compulsory misses [1]. Hardware prefetching scheme adds a hardware module to the processor. This module will monitor the behaviour of programs and predicts their next misses by assuming a predefined access pattern. Hardware prefetching schemes basically are divided into two categories: global predictors and local predictors.

Global predictors consider all misses in one stream and based on this the prediction of next misses' takes place. Context-based predictors [2–3] and & early prefetching schemes such as sequential prefetching [4] lie in this category.

A. Low Power L1 Data Cache

For a high capacity and data sharing, modern MPSoCs prefer large L2 caches. To achieve a low miss ratio, they employ set-associative caches. However, the set-associative caches are energy inefficient because the requested data has only one way and other ways are merely necessary to be accessed. To enhance the energy efficiency of set-associative caches, way-predicting L1 caches have been proposed [5]. They remember that the MRU way number for each set and the way is accessed first in the next L1 cache access. The MRU heuristic works fine for L1 caches because the L1 caches will have a high degree of locality. However, in L2 caches, most of accesses with the high locality are filtered by L1 caches, thus making the MRU heuristic less efficient.

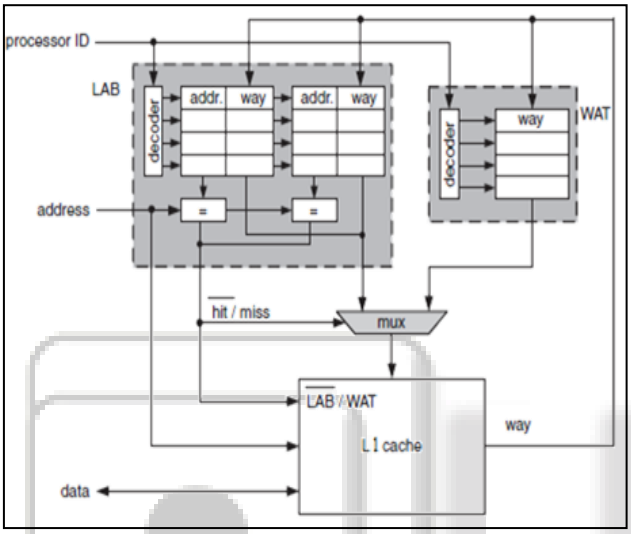

Fig. 1: Logic diagram oflow power way-predicting L1 cache (with 2-way LAB and WAT) for 4-way MPSoC.

B. Cache Miss Penalty

As the CPU clock rates have increased at a much faster rate than DRAM, the relative cost of miss penalties has actually increased over time. Wide gap b/w CPU and MM access time might lead to a catch,

- Fast Cache → small size → low number of hits.
- Large Cache → slow access time → CPU stalls.

As the L1 designed to minimize hit time, so implication on this is local miss rate which can b given as:

- Local (cache) Miss Rate = Number of Misses at Cache Level/Total Number of CPU memory references generated
- Consider the Average Memory Access Time for L1 Cache = Hit Time(L1) + Miss Rate(L1) × Miss Penalty(L1)

Therefore the miss penalty expressed in terms:

- Miss Penalty (L1) = Hit Time(L2) + Miss Rate(L2) × Miss Penalty(L2)

So we can find the global miss rate as follows:

- Global (cache) Miss Rate =Number of Misses up to specified cache level / Total Number of CPU memory references generated

III. USING HEURISTIC OF LOW POWER L1 CACHE

For the low power L1 cache shown in the figure 1 we can apply the heuristics by using the Genetic Algorithms.

A. Genetic Algorithm

Genetic algorithm is a method for an effective search in a vast area which at last creates a way to get a best answer [6]. Genetic algorithm is different from the previous ones. Such algorithms designing are working with some encoded variables. The advantage of that is codes are able to convert continual areas to disconnected ones. One of the differences between the GA with preview ones is that in GA we dead; with a population or set of points at once. Which in the old methods we operated just with the one special point. It means that GA can cover a lot of designs at once another point is that GA is based on the accidental processing. Generally in using GA three important issues should be clarified here [6].

- Definition of the aim function or expense function.
- Definition and implementation of the genetic area.
- Definition and implementation of the GA operations.

As long as these three parts are well defined the GA will work correctly and overall system's efficiency can be improved by doing source changes GA is designed on the base of best species survival and best kind's reproduction.

First the algorithm starts by a set of accidental responses i.e. chromosomes which called population. These responses population method propulsion. These responses are used or build method for choosing new population are depend on their stability so the best ones have more chance to reproduce. This continues to come to final criteria (access to the best solution).

B. Rules

Each MU in mobile Database is connected to one of the MSS in order to using the data in Databases. Also it gives its request and receives the essential data from them which are done through wireless network. In this design the main Database dispatched on the MSS each mobile unit has some design as the main Database which is used to cache the data and accepting local data. Every MSS is responsible for making connection with MU's. MSS are also responsible for accuracy of transaction and registration of the transactions results in Databases. In a connection between MU & Mss. MSS is responsible for creating an agent to control the transactions and manage displace mint MU's. In fact there is an agent for each MU and existence of any of the agent in MSS is a sign of the MU connected to MSS.

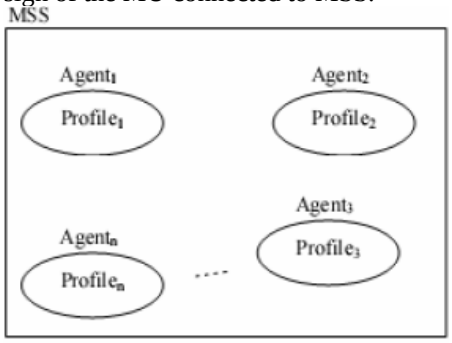

Fig. 2: agents in MSS

C. Definition

1) Population

All agents in MSS connected to Fixed or mobile network.

2) Chromosome

We consider that each agent as a chromosome.

3) Gene
The records on each agent's table. Its features are some like index structure.
4) Crossover
Finding one of the best data (records) for predicting and sending to the MU.
5) Objective function
A function to help us choose records

D. Encoding and decoding of genetic

For predicting records to send to MU we need decoding of Genetic that is done as follow:

Some genes (record) which is requested by the MU should be set fixed in the chromosome. Note that at the end of given algorithm and by having the new chromosome, the above encoding should be decoded. Each gene contains two table codes and the records. Using these codes, we should find the selected record and put in sending pack. It should be done for all the chromosomes then send the packet to MU.

E. Cache Memory Power Reduction Techniques Clock distribution throughout the system

Cache is a major source of power consumption in the memory system. Power saving techniques can be performed in following gate level: Low-level gating, Mid-level gating, and High-level gating. Low-level gating concentrates only the registers that are needed for next stage. Mid-level gating is mainly on the precharge load of a cache during cache miss. Thus the proposed research is on the high-level gating where the

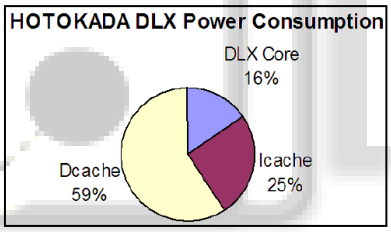

Fig. 3: Division of Power Consumption per Components of the HOTOKADA

Top-level clock is wired to minimize switching of the whole cache for power reduction.

Independent of the configuration, high-level gating provides further energy savings beyond those obtained by the mid-level gating without performance penalty. [7] This is also one of the main reason why the focus of this research is on high-level power reduction techniques. The following sections will discuss the existing high-level gating power reduction techniques.

IV. HEURISTIC PREDICTION FOR L1 DATA CACHE MISSES FOR LOW POWER CACHE MEMORY & REDUCING THE CACHE MISS PENALTY

A. Prediction

When MU offer a request but the necessary data are not in the MU, predict is done. In this case, to get the data, MU sends its request to MSS. The chromosome in the MSS should not also send necessary data for firm genes but also predict another data and send them to MU for doing predication regarding to gene data. One should compute the record code (RC) and the table code (TC) to the line with the highest Vote. We should omit the genes in the chromosome which are asked, since they settled fixed in new Crossover chromosomes. Then according to the genetic algorithm, the genes with the best selection amount are chosen and added to new chromosome and created new generation. After decoding, records are sending to MU.

B. Applying prediction with genetic algorithm to reduce cache miss penalty

1) Choosing some of the chromosomes between all for preliminary population
2) Appointing the selection function which is computed through commutation of Votes.
3) Producing the new chromosome by:
   – Choosing the two chromosomes from population and according to Objective function for Crossover.
   – Merging the two parental chromosomes with one possibility in order to get the new generation.
4) Using new population for reccomplishment.
5) If there is not any final condition (ceasing improvement

Figure 4 shows the architecture of the whole system on which heuristic is applied.

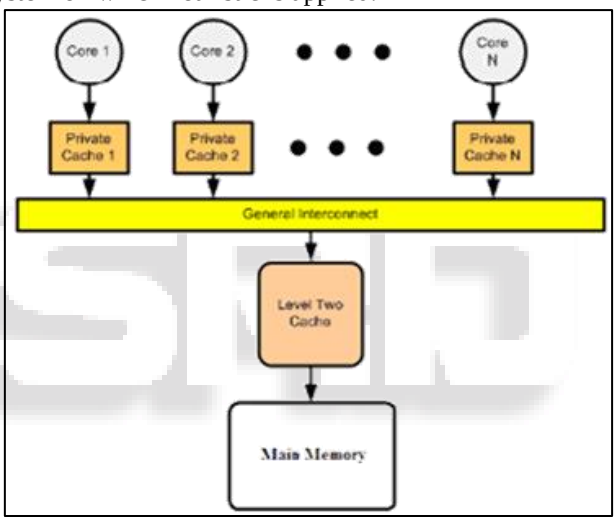

Fig. 4: Architecture of the Whole System

C. Prediction with Genetic Algorithm

1) Choosing some of the chromosomes between all for preliminary population
2) Appointing selection function which is computed through the commutation of Votes.
3) Producing the new chromosome by:
   – Choosing the two chromosomes from population and according to Objective function for Crossover.
   – Merging the two parental chromosomes with one possibility in order to get the new generation.
4) Using the new population for the reccomplishment.
5) if there is not any final condition (ceasing improvement

Appling above heuristic on the reducing cache miss penalty we use Non-blocking caches – stall reduction on cache misses**.** Out of the order execution and compiler optimization for scheduling loads away from operand use imply that operands should not stall on a cache miss. Non blocking (lockup free) caches will support hits on instructions issued after a cache miss. Cache controller complexity increases, required to track the multiple misses and hits points shown as,

- FP programs benefits from supporting higher depths of hit under miss.
- INT programs gains most from a single hit under miss.

D. Implementing the Heuristic in Hardware

We could implement our cache tuning heuristic in either the software or hardware. However, in a software-based approach, system processor would execute the search heuristic.

Executing the heuristic on system processor would not only change the runtime behavior of the application but also affect the cache behavior, possibly resulting in search heuristic choosing a non-optimal cache configuration. Therefore, a hardware based approach that does not significantly impact system's area or the power consumption is more desirable. We implemented search heuristic in hardware using a simple state machine controlling a simple data path, shown in the Figure 6. In data path, there are eighteen registers.

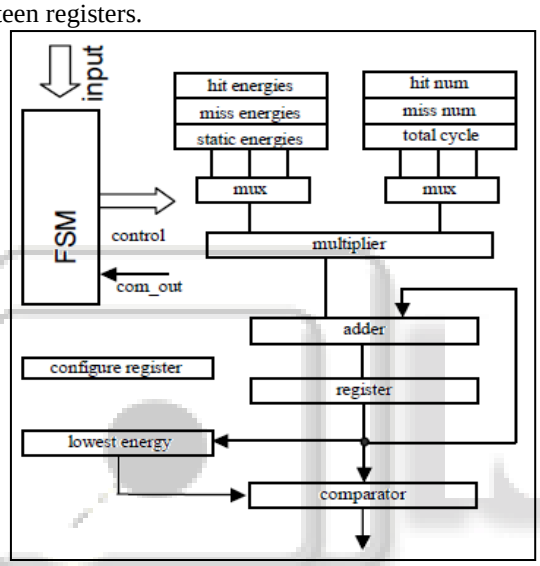

Fig. 6: FSM+D of cache tuner (The "input" includes clock, reset, and start signal. "control" is the output of FSM to control the registers and muxes; the output of the comparator is fed back to FSM).

We use three of the registers to collect the runtime information, total cache hits, total cache misses, and total cycles. Six additional registers store cache hit energy per cache access, which is correspond to 8 Kbytes 4-way, 2-way, and 1-way; 4Kbytes 2-way and 1-way; and 2 Kbytes 1-way configurations. The physical line size is 16 bytes, so cache hit energy for different cache line sizes is the same. We use the three registers to store the miss energy, which corresponds to line sizes of 16 bytes, 32 bytes, and 64 bytes respectively. Because the static power dissipation depends on the cache size only, we use the three more registers to store the static power dissipation corresponding to 8 Kbyte, 4 Kbyte, and 2 Kbyte caches, respectively. All the fifteen registers are 16 bits wide. We also need one register to hold the result of the energy calculations and another register to hold the lowest energy of the cache configurations tested. Both these registers are 32 bits wide. The last register is the configure register that is used to configure cache. We have four cache parameters to configure, where cache size, line size and the associativity have three possible values, and prediction can either be on or off. Therefore, the configure register is seven bits wide. The FSM controls the datapath using signal "control" and the output of comparator within the data path is an input to the FSM.

V. EXPERIMENT RESULTS AND EVALUATION

For the Suggested algorithm test, we created a program which is assimilated in MSS and MU environment. In addition, we have created a fixed Database and we created a table in the MU consisted of 400 quarries from Database in the MSS. In this program the MU created a random quarry from the above mentioned table. The selected quarry received by the MU process unit. If the intended information cannot be found in MU buffer, the selected quarry is sent to MSS in order to have a response. And there is another process which calculated all the received quarries from MU and also all received information from MSS in buffer 40 modes which are shown is the following:

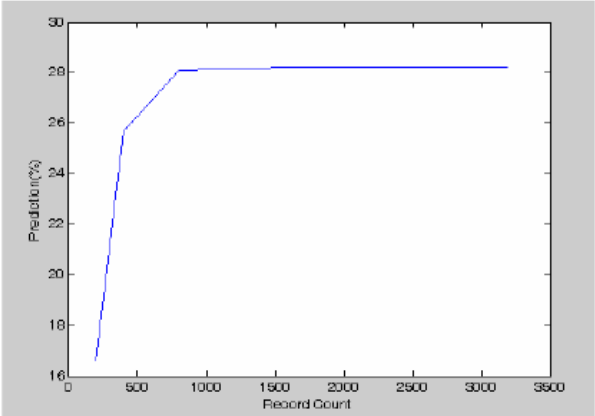

Fig. 7: Prediction with 20 Buffers in MU

The effective access time. The base bar shows the L1 data cache size of the base system. The base system has no improvements in its memory system. The other bars show the access time for each prefetching scheme. Based on the average bar, it can be seen that the L1 data access time of the base system is around 91.8%. The effective access time of Markov is 92.7% and that of stream buffers is 92.9%, whereas it is 96.8% for the TPP. In other words, the stream buffers and Markov approaches increase the effective access time only by 1%, whereas the TPP increases it by 5%. In other words, the effective access time improves five times better than other approaches in the above algorithm.

VI. CONCLUSION AND FUTURE SCOPE

Identifying the problem of the miss penalty, this paper suggested an elegant solution. Measurements showed that the approach improves configurable cache enables tuning of the cache to a particular program, which can significantly reduce the memory access power that often accounts for half a microprocessor system's power. Our self-tuning on-chip CAD method relieves the designers from the burden of having to perform simulations or manual physical measurements to determine the best configuration, finding best configuration automatically. Our heuristic minimizes the number of configurations examined during the tuning, and minimizes the need for cache flushing. Energy savings of such a cache average 40% compared to a the standard cache. The self-tuning cache can be used in a variety of approaches.

In present research central attention is on data which are needed now, so just that data are cached. There is not any prediction for possible needed data. In this research, genetic algorithm, which is the best one and has an appropriate turn over in an appointed time, is used.

Fuzzy logic, neuron nets and improvement algorithm can be used in this prediction.

REFERENCE

[1] Jouppi, N.P.: 'Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers'ISCA-17, May 1990, pp. 364–373.
[2] Charney, M.J., and Puzak, T.R.: 'Prefetching and memory system behavior of the SPEC95 benchmark suite', IBM J. Res. Dev., 1997, 41, pp. 265–286.
[3] Mahjur, A., Jahangir, A.H., and Gholamipour, A.H.: 'On the performance of trace locality of reference', Perform. Eval., 2005, 60, (1–4), pp. 51–72.
[4] Vanderwiel, S.P., and Lilja, D.J.: 'Data prefetch mechanisms', ACM Comput. Surv., 2000, 32, (2), pp. 174–199.
[5] Inoue, K., Ishihara, T., and Murakami, K.: 'Way-predicting setassociative cache for high performance and low energy consumption'. Proc. of ISLPED, San Diego, CA, USA, 1999, pp. 273–275.
[6] Alireza, Mahdi, genetic algorithms and its application, Yazd university of iran,2006.
[7] Duarte, D.; Narayanan, V.; Irwin, M.J.; Kandemir, M., "Evaluating the impact of architectural-level optimizations on clock power," 14th Annual IEEE International ASIC/SOC Conference, Proceedings, 2001.
[8] Chakrabarti, C., ``Cache design and exploration for low power embedded systems", IEEE International Conference on Performance, Computing, and Communications, 2001.
[9] Dioquino, et. al., ``DLX HOTOKADA: A Design and Implementation of a 32-Bit Dual Core Capable DLX Microprocessor with Single Level Cache," The 15th IEEE International Conference on Electronics, Circuits and Systems, 2008.
[10] AGARWAL. A., HOROWITZ, M., AND HENNESSY. J., 1989. An Analytical Cache Model. ACM Transactions on Computer Systems, 7(2): 184-215.

